Rtips. Revival 2014!

Paul E. Johnson <pauljohn @ ku.edu>

March 24, 2014

The original Rtips started in 1999. It became difficult to update because of limitations in the software with which it was created. Now I know more about R, and have decided to wade in again. In January, 2012, I took the FaqManager HTML output and converted it to LaTeX with the excellent open source program pandoc, and from there I've been editing and updating it in LyX.

From here on out, the latest html version will be at `http://pj.freefaculty.org/R/Rtips.html` and the PDF for the same will be

`http://pj.freefaculty.org/R/Rtips.pdf`.

You are reading the New Thing!

The first chore is to cut out the old useless stuff that was no good to start with, correct mistakes in translation (the quotation mark translations are particularly dangerous, but also there is trouble with ~, $, and -.

Original Preface

(I thought it was cute to call this "StatsRus" but the Toystore's lawyer called and, well, you know. . .)

If you need a tip sheet for R, here it is.

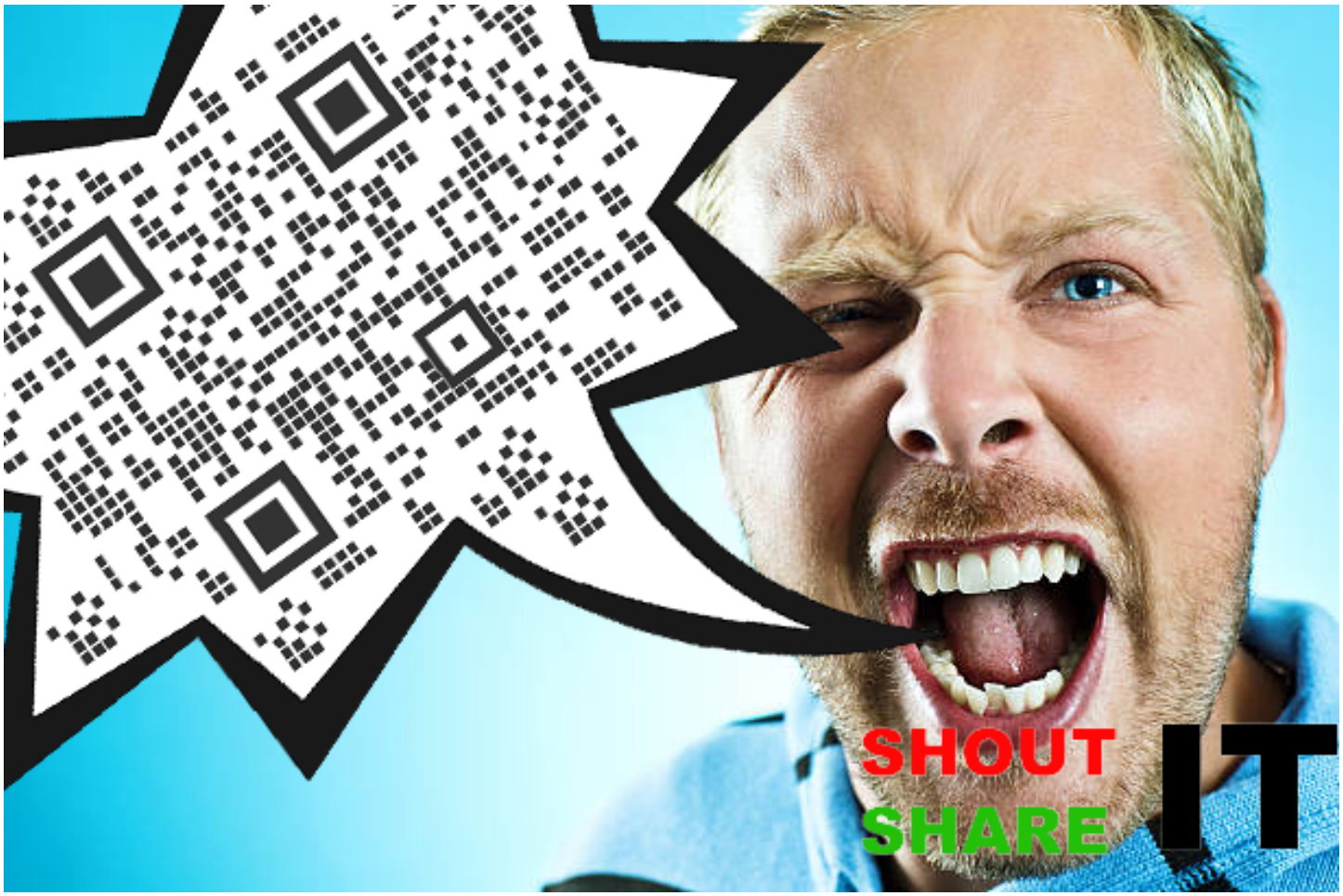
This is not a substitute for R documentation, just a list of things I had trouble remembering when switching from SAS to R.

Heed the words of Brian D. Ripley, "One enquiry came to me yesterday which suggested that some users of binary distributions do not know that R comes with two Guides in the doc/manual directory plus an FAQ and the help pages in book form. I hope those are distributed with all the binary distributions (they are not made nor installed by default). Windows-specific versions are available." Please run "help.start()" in R!

Contents

1 Data Input/Output **5**

1.1 Bring raw numbers into R (05/22/2012) 5
1.2 Basic notation on data access (12/02/2012) 6
1.3 Checkout the new Data Import/Export manual (13/08/2001) 6
1.4 Exchange data between R and other programs (Excel, etc) (01/21/2009) 6
1.5 Merge data frames (04/23/2004) 7
1.6 Add one row at a time (14/08/2000) 9
1.7 Need yet another different kind of merge for data frames (11/08/2000) 9
1.8 Check if an object is NULL (06/04/2001) 10


SHOUT
SHARE
IT

1.9 Generate random numbers (12/02/2012) . . . 10
1.10 Generate random numbers with a fixed mean/variance (06/09/2000) . . . 11
1.11 Use rep to manufacture a weighted data set (30/01/2001) . . . 11
1.12 Convert contingency table to data frame (06/09/2000) . . . 12
1.13 Write: data in text file (31/12/2001) . . . 12

2 Working with data frames: Recoding, selecting, aggregating 12
2.1 Add variables to a data frame (or list) (02/06/2003) . . . 12
2.2 Create variable names "on the fly" (10/04/2001) . . . 13
2.3 Recode one column, output values into another column (12/05/2003) . . . 13
2.4 Create indicator (dummy) variables (20/06/2001) . . . 16
2.5 Create lagged values of variables for time series regression (05/22/2012) . . . 16
2.6 How to drop factor levels for datasets that don't have observations with those values? (08/01/2002) . . . 17
2.7 Select/subset observations out of a dataframe (08/02/2012) . . . 17
2.8 Delete first observation for each element in a cluster of observations (11/08/2000) 18
2.9 Select a random sample of data (11/08/2000) . . . 18
2.10 Selecting Variables for Models: Don't forget the subset function (15/08/2000) . . 19
2.11 Process all numeric variables, ignore character variables? (11/02/2012) . . . 19
2.12 Sorting by more than one variable (06/09/2000) . . . 19
2.13 Rank within subgroups defined by a factor (06/09/2000) . . . 20
2.14 Work with missing values (na.omit, is.na, etc) (15/01/2012) . . . 20
2.15 Aggregate values, one for each line (16/08/2000) . . . 21
2.16 Create new data frame to hold aggregate values for each factor (11/08/2000) . . 21
2.17 Selectively sum columns in a data frame (15/01/2012) . . . 21
2.18 Rip digits out of real numbers one at a time (11/08/2000) . . . 21
2.19 Grab an item from each of several matrices in a List (14/08/2000) . . . 22
2.20 Get vector showing values in a dataset (10/04/2001) . . . 22
2.21 Calculate the value of a string representing an R command (13/08/2000) . . . 22
2.22 "Which" can grab the index values of cases satisfying a test (06/04/2001) . . . 22
2.23 Find unique lines in a matrix/data frame (31/12/2001) . . . 23

3 Matrices and vector operations 23
3.1 Create a vector, append values (01/02/2012) . . . 23
3.2 How to create an identity matrix? (16/08/2000) . . . 24
3.3 Convert matrix "m" to one long vector (11/08/2000) . . . 24
3.4 Creating a peculiar sequence (1 2 3 4 1 2 3 1 2 1) (11/08/2000) . . . 24
3.5 Select every n'th item (14/08/2000) . . . 25
3.6 Find index of a value nearest to 1.5 in a vector (11/08/2000) . . . 25
3.7 Find index of nonzero items in vector (18/06/2001) . . . 25
3.8 Find index of missing values (15/08/2000) . . . 26
3.9 Find index of largest item in vector (16/08/2000) . . . 26
3.10 Replace values in a matrix (22/11/2000) . . . 26
3.11 Delete particular rows from matrix (06/04/2001) . . . 27
3.12 Count number of items meeting a criterion (01/05/2005) . . . 27
3.13 Compute partial correlation coefficients from correlation matrix (08/12/2000) . . 27
3.14 Create a multidimensional matrix (R array) (20/06/2001) . . . 28
3.15 Combine a lot of matrices (20/06/2001) . . . 28
3.16 Create "neighbor" matrices according to specific logics (20/06/2001) . . . 28
3.17 "Matching" two columns of numbers by a "key" variable (20/06/2001) . . . 29
3.18 Create Upper or Lower Triangular matrix (06/08/2012) . . . 29
3.19 Calculate inverse of X (12/02/2012) . . . 30

3.20 Interesting use of Matrix Indices (20/06/2001) . . . 31
3.21 Eigenvalues example (20/06/2001) . . . 31

4 Applying functions, tapply, etc **32**
4.1 Return multiple values from a function (12/02/2012) . . . 32
4.2 Grab "p" values out of a list of significance tests (22/08/2000) . . . 32
4.3 ifelse usage (12/02/2012) . . . 32
4.4 Apply to create matrix of probabilities, one for each "cell" (14/08/2000) . . . 32
4.5 Outer. (15/08/2000) . . . 33
4.6 Check if something is a formula/function (11/08/2000) . . . 33
4.7 Optimize with a vector of variables (11/08/2000) . . . 33
4.8 slice.index, like in S+ (14/08/2000) . . . 33

5 Graphing **33**
5.1 Adjust features with par before graphing (18/06/2001) . . . 33
5.2 Save graph output (03/21/2014) . . . 34
5.3 How to automatically name plot output into separate files (10/04/2001) . . . 36
5.4 Control papersize (15/08/2000) . . . 36
5.5 Integrating R graphs into documents: LaTeX and EPS or PDF (20/06/2001) . . . 37
5.6 "Snapshot" graphs and scroll through them (31/12/2001) . . . 37
5.7 Plot a density function (eg. Normal) (22/11/2000) . . . 37
5.8 Plot with error bars (11/08/2000) . . . 37
5.9 Histogram with density estimates (14/08/2000) . . . 37
5.10 How can I "overlay" several line plots on top of one another? (09/29/2005) . . . 37
5.11 Create "matrix" of graphs (18/06/2001) . . . 39
5.12 Combine lines and bar plot? (07/12/2000) . . . 39
5.13 Regression scatterplot: add fitted line to graph (03/20/2014) . . . 40
5.14 Control the plotting character in scatterplots? (11/08/2000) . . . 40
5.15 Scatterplot: Control Plotting Characters (men vs women, etc)} (11/11/2002) . . . 41
5.16 Scatterplot with size/color adjustment (12/11/2002) . . . 41
5.17 Scatterplot: adjust size according to 3rd variable (06/04/2001) . . . 42
5.18 Scatterplot: smooth a line connecting points (02/06/2003) . . . 42
5.19 Regression Scatterplot: add estimate to plot (18/06/2001) . . . 42
5.20 Axes: controls: ticks, no ticks, numbers, etc (22/11/2000) . . . 42
5.21 Axes: rotate labels (06/04/2001) . . . 43
5.22 Axes: Show formatted dates in axes (06/04/2001) . . . 43
5.23 Axes: Reverse axis in plot (12/02/2012) . . . 43
5.24 Axes: Label axes with dates (11/08/2000) . . . 44
5.25 Axes: Superscript in axis labels (11/08/2000) . . . 44
5.26 Axes: adjust positioning (31/12/2001) . . . 44
5.27 Add "error arrows" to a scatterplot (30/01/2001) . . . 44
5.28 Time Series: how to plot several "lines" in one graph? (06/09/2000) . . . 45
5.29 Time series: plot fitted and actual data (11/08/2000) . . . 45
5.30 Insert text into a plot (22/11/2000) . . . 45
5.31 Plotting unbounded variables (07/12/2000) . . . 45
5.32 Labels with dynamically generated content/math markup (16/08/2000) . . . 45
5.33 Use math/sophisticated stuff in title of plot (11/11/2002) . . . 46
5.34 How to color-code points in scatter to reveal missing values of 3rd variable? (15/08/2000) . . . 46
5.35 lattice: misc examples (12/11/2002) . . . 46
5.36 Make 3d scatterplots (11/08/2000) . . . 46
5.37 3d contour with line style to reflect value (06/04/2001) . . . 47

5.38 Animate a Graph! (13/08/2000) . . . 47
5.39 Color user-portion of graph background differently from margin (06/09/2000) . . 47
5.40 Examples of graphing code that seem to work (misc) (11/16/2005)} . . . 48

6 Common Statistical Chores **51**
6.1 Crosstabulation Tables (01/05/2005) . . . 51
6.2 t-test (18/07/2001) . . . 51
6.3 Test for Normality (31/12/2001) . . . 52
6.4 Estimate parameters of distributions (12/02/2012) . . . 52
6.5 Bootstrapping routines (14/08/2000) . . . 52
6.6 BY subgroup analysis of data (summary or model for subgroups)(06/04/2001) . 52

7 Model Fitting (Regression-type things) **53**
7.1 Tips for specifying regression models (12/02/2002) . . . 53
7.2 Summary Methods, grabbing results inside an "output object" . . . 53
7.3 Calculate separate coefficients for each level of a factor (22/11/2000) . . . 53
7.4 Compare fits of regression models (F test subset B's =0) (14/08/2000) . . . 54
7.5 Get Predicted Values from a model with predict() (11/13/2005) . . . 55
7.6 Polynomial regression (15/08/2000) . . . 57
7.7 Calculate "p" value for an F stat from regression (13/08/2000) . . . 57
7.8 Compare fits (F test) in stepwise regression/anova (11/08/2000) . . . 57
7.9 Test significance of slope and intercept shifts (Chow test?) . . . 58
7.10 Want to estimate a nonlinear model? (11/08/2000) . . . 58
7.11 Quasi family and passing arguments to it. (12/11/2002) . . . 58
7.12 Estimate a covariance matrix (22/11/2000) . . . 58
7.13 Control number of significant digits in output (22/11/2000) . . . 59
7.14 Multiple analysis of variance (06/09/2000) . . . 59
7.15 Test for homogeneity of variance (heteroskedasticity) (12/02/2012) . . . 59
7.16 Use nls to estimate a nonlinear model (14/08/2000) . . . 60
7.17 Using nls and graphing things with it (22/11/2000) . . . 60
7.18 $-2Log(L)$ and hypo tests (22/11/2000) . . . 60
7.19 logistic regression with repeated measurements (02/06/2003) . . . 61
7.20 Logit (06/04/2001) . . . 61
7.21 Random parameter (Mixed Model) tips (01/05/2005) . . . 61
7.22 Time Series: basics (31/12/2001) . . . 61
7.23 Time Series: misc examples (10/04/2001) . . . 62
7.24 Categorical Data and Multivariate Models (04/25/2004) . . . 62
7.25 Lowess. Plot a smooth curve (04/25/2004) . . . 62
7.26 Hierarchical/Mixed linear models. (06/03/2003) . . . 62
7.27 Robust Regression tools (07/12/2000) . . . 63
7.28 Durbin-Watson test (10/04/2001) . . . 63
7.29 Censored regression (04/25/2004) . . . 63

8 Packages **63**
8.1 What packages are installed on Paul's computer? . . . 63
8.2 Install and load a package . . . 65
8.3 List Loaded Packages . . . 65
8.4 Where is the default R library folder? Where does R look for packages in a computer? . . . 66
8.5 Detach libraries when no longer needed (10/04/2001) . . . 66

9 Misc. web resources **66**
9.1 Navigating R Documentation (12/02/2012) . . . 66
9.2 R Task View Pages (12/02/2012) . . . 66
9.3 Using help inside R(13/08/2001) . . . 67
9.4 Run examples in R (10/04/2001) . . . 67

10 R workspace **67**
10.1 Writing, saving, running R code (31/12/2001) . . . 67
10.2 .RData, .RHistory. Help or hassle? (31/12/2001) . . . 68
10.3 Save & Load R objects (31/12/2001) . . . 68
10.4 Reminders for object analysis/usage (11/08/2000) . . . 68
10.5 Remove objects by pattern in name (31/12/2001) . . . 68
10.6 Save work/create a Diary of activity (31/12/2001) . . . 69
10.7 Customized Rprofile (31/12/2001) . . . 69

11 Interface with the operating system **69**
11.1 Commands to system like "change working directory" (22/11/2000) . . . 69
11.2 Get system time. (30/01/2001) . . . 69
11.3 Check if a file exists (11/08/2000) . . . 69
11.4 Find files by name or part of a name (regular expression matching) (14/08/2001) 70

12 Stupid R tricks: basics you can't live without **70**
12.1 If you are asking for help (12/02/2012) . . . 70
12.2 Commenting out things in R files (15/08/2000) . . . 71

13 Misc R usages I find interesting **71**
13.1 Character encoding (01/27/2009) . . . 71
13.2 list names of variables used inside an expression (10/04/2001) . . . 71
13.3 R environment in side scripts (10/04/2001) . . . 71
13.4 Derivatives (10/04/2001) . . . 72

1 Data Input/Output

1.1 Bring raw numbers into R (05/22/2012)

This is truly easy. Suppose you've got numbers in a space-separated file "myData", with column names in the first row (thats a header). Run

```
myDataFrame <- read.table(``myData'',header=TRUE)
```

If you type "?read.table" it tells about importing files with other delimiters.

Suppose you have tab delimited data with blank spaces to indicate "missing" values. Do this:

```
myDataFrame<-read.table("myData",sep="\t",na.strings=" ",header=TRUE)
```

Be aware than anybody can choose his/her own separator. I am fond of "|" because it seems never used in names or addresses (unlike just about any other character I've found).

Suppose your data is in a compressed gzip file, myData.gz, use R's gzfile function to decompress on the fly. Do this:

```
myDataFrame <- read.table(gzfile("myData.gz"), header=TRUE)
```

If you read columns in from separate files, combine into a data frame as:

```
variable1 <- scan("file1")
variable2 <- scan("file2")
mydata <- cbind(variable1, variable2)
#or use the equivalent:
#mydata <- data.frame(variable1, variable2)
#Optionally save dataframe in R object file with:
write.table(mydata, file="filename3")
```

1.2 Basic notation on data access (12/02/2012)

To access the columns of a data frame "x" with the column number, say x[,1], to get the first column. If you know the column name, say "pjVar1", it is the same as x$pjVar1 or x[, "pjVar1"].

Grab an element in a list as x[[1]]. If you just run x[1] you get a list, in which there is a single item. Maybe you want that, but I bet you really want x[[1]]. If the list elements are named, you can get them with x$pjVar1 or x[["pjVar1"]].

For instance, if a data frame is "nebdata" then grab the value in row 1, column 2 with:

```
nebdata[1, 2]
## or to selectively take from column2 only when the column Volt equals "ABal"
nebdata[nebdata$Volt=="ABal", 2]
(from Diego Kuonen)
```

1.3 Checkout the new Data Import/Export manual (13/08/2001)

With R–1.2, the R team released a manual about how to get data into R and out of R. That's the first place to look if you need help. It is now distributed with R. Run

```
help.start()
```

1.4 Exchange data between R and other programs (Excel, etc) (01/21/2009)

Patience is the key. If you understand the format in which your data is currently held, chances are good you can translate it into R with more or less ease.

Most commonly, people seem to want to import Microsoft Excel spreadsheets. I believe there is an ODBC approach to this, but I think it is only for MS Windows.

In the gdata package, which was formerly part of gregmisc, there is a function that can use Perl to import an Excel spreadsheet. If you install gdata, the function to use is called "read.xls". You have to specify which sheet you want. That works well enough if your data is set in a numeric format inside Excel. If it is set with the GENERAL type, I've seen the imported numbers turn up as all asterixes.

Other packages have appeared to offer Excel import ability, such as xlsReadWrite.

In the R-help list, I also see reference to an Excel program addon called RExcel that can install an option in the Excel menus called "Put R Dataframe". The home for that project is http://sunsite.univie.ac.at/rcom/

I usually proceed like this.

Step 1. Use Excel to edit the sheet so that it is RECTANGULAR. It should have variable names in row 1, and it has numbers where desired and the string NA otherwise. It must have NO embedded formulae or other Excel magic. Be sure that the columns are declared with the proper Excel format. Numerical information should have a numerical type, while text should have text as the type. Avoid "General".

Step 2. First, try the easy route. Try gdata's "read.xls" method. As long as you tell it which sheet you want to read out of your data set, and you add header=T and whatever other options you'd like in an ordinary read.table usage, then it has worked well for us.

Step 3. Suppose step 2 did not work. Then you have something irregular in your Excel sheet and you should proceed as follows. Either open Excel and clean up your sheet and try step 2 again, or take the more drastic step: Export the spread sheet to text format. File/Save As, navigate to "csv" or "txt" and choose any options like "advanced format" or "text configurable". Choose the delimiter you want. One option is the tab key. When I'm having trouble, I use the bar symbol, "|", because there's little chance it is in use inside a column of data. If your columns have addresses or such, usage of the COMMA as a delimiter is very dangerous.

After you save that file in text mode, open it in a powerful "flat text" editor like Emacs and look through to make sure that 1) your variable names in the first row do not have spaces or quotation marks or special characters, and 2) look to the bottom to make sure your spreadsheet program did not insert any crud. If so, delete it. If you see commas in numeric variables, just use the Edit/Search/replace option to delete them.

Then read in your tab separated data with

```
read.table(``filename'',header=T,sep=``\t'')
```

I have tested the "foreign" package by importing an SPSS file and it worked great. I've had great results importing Stata Data sets too.

Here's a big caution for you, however. If your SPSS or Stata numeric variables have some "value lables", say 98=No Answer and 99=Missing, then R will think that the variable is a factor, and it will convert it into a factor with 99 possible values. The foreign library commands for reading spss and dta files have options to stop R from trying to help with factors and I'd urge you to read the manual and use them.

If you use read.spss, for example, setting use.value.labels=F will stop R from creating factors at all. If you don't want to go that far, there's an option max.value.labels that you can set to 5 or 10, and stop it from seeing 98=Missing and then creating a factor with 98 values. It will only convert variables that have fewer than 5 or 10 values. If you use read.dta (for Stata), you can use the option convert.factors=F.

Also, if you are using read.table, you may have trouble if your numeric variables have any "illegal" values, such as letters. Then R will assume you really intend them to be factors and it will sometimes be tough to fix. If you add the option as.is=T, it will stop that cleanup effort by R.

At one time, the SPSS import support in foreign did not work for me, and so I worked out a routine of copying the SPSS data into a text file, just as described for Excel.

I have a notoriously difficult time with SAS XPORT files and don't even try anymore. I've seen several email posts by Frank Harrel in r-help and he has some pretty strong words about it. I do have one working example of importing the Annenberg National Election Study into R from SAS and you can review that at http://pj.freefaculty.org/DataSets/ANES/2002. I wrote a long boring explanation. Honestly, I think the best thing to do is to find a bridge between SAS and R, say use some program to convert the SAS into Excel, and go from there. Or just write the SAS data set to a file and then read into R with read.table() or such.

1.5 Merge data frames (04/23/2004)

update:Merge is confusing! But if you study this, you will see everything in perfect clarity:

```
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
```

```
x4 <- rnorm(100)
ds1 <- data.frame(city=rep(1,100),x1=x1,x2=x2)
ds2 <- data.frame(city=rep(2,100),x1=x1,x3=x3,x4=x4)
merge(ds1,ds2,by.x=c(``city'',``x1''),by.y=c(``city'',``x1''),all=TRUE)
```

The trick is to make sure R understands which are the "common variables" in the two datasets so it lines them up, and then all=T is needed to say that you don't want to throw away the variables that are only in one set or the other. Read the help page over and over, you eventually get it.

More examples:

```
experiment <- data.frame(times = c(0,0,10,10,20,20,30,30), expval = c
    (1,1,2,2,3,3,4,4))
simul <- data.frame(times = c(0,10,20,30), simul = c(3,4,5,6))
```

I want a merged datatset like:

```
  times expval simul
1 0 1 3
2 0 1 3
3 10 2 4
4 10 2 4
5 20 3 5
6 20 3 5
7 30 4 6
8 30 4 6
```

Suggestions

```
merge(experiment, simul)
(from Brian D. Ripley)
```

does all the work for you.

Or consider:

```
exp.sim <- data.frame(experiment,simul=simul$simul[match(experiment$times,simul$
    times)])
(from Jim Lemon)
```

I have dataframes like this:

```
 state count1 percent1
CA 19 0.34
TX 22 0.35
FL 11 0.24
OR 34 0.42
GA 52 0.62
MN 12 0.17
NC 19 0.34
state count2 percent2
FL 22 0.35
MN 22 0.35
CA 11 0.24
TX 52 0.62
```

And I want

```
state count1 percent1 count2 percent2
CA 19 0.34 11 0.24
TX 22 0.35 52 0.62
FL 11 0.24 22 0.35
OR 34 0.42 0 0
GA 52 0.62 0 0
```

```
MN 12 0.17 22 0.35
NC 19 0.34 0 0
( from Yu-Ling Wu )
```

In response, Ben Bolker said

```
state1 <-
c(``CA'',``TX'',``FL'',``OR'',``GA'',``MN'',``NC'')
count1 <- c(19,22,11,34,52,12,19)
percent1 <- c(0.34,0.35,0.24,0.42,0.62,0.17,0.34)
state2 <- c(``FL'',``MN'',``CA'',``TX'')
count2 <- c(22,22,11,52)
percent2 <- c(0.35,0.35,0.24,0.62)
data1 <- data.frame(state1,count1,percent1)
data2 <- data.frame(state2,count2,percent2)

datac <- data1m <- match(data1$state1,data2$state2,0)
datac$count2 <- ifelse(m<-0,0,data2$count2[m])
datac$percent2 <- ifelse(m<-0,0,data2$percent2[m])
```

If you didn't want to keep all the rows in both data sets (but just the shared rows) you could use

```
merge(data1,data2,by=1)
```

1.6 Add one row at a time (14/08/2000)

Question: I would like to create an (empty) data frame with "headings" for every column (column titles) and then put data row-by-row into this data frame (one row for every computation I will be doing), i.e.

```
no. time temp pressure <---the headings
 1    0     100    80     <---first result
 2    10    110    87     <---2nd result .....
```

Answer: Depends if the cols are all numeric: if they are a matrix would be better. But if you insist on a data frame, here goes:

If you know the number of results in advance, say, N, do this

```
df <- data.frame(time=numeric(N), temp=numeric(N), pressure=numeric(N))
df[1, ] <- c(0, 100, 80)
df[2, ] <- c(10, 110, 87)
...
```

or

```
m <- matrix(nrow=N, ncol=3)
colnames(m) <- c("time", "temp", "pressure")
m[1, ]  <- c(0, 100, 80)
m[2, ] <- c(10, 110, 87)
```

The matrix form is better size it only needs to access one vector (a matrix is a vector with attributes), not three.

If you don't know the final size you can use rbind to add a row at a time, but that is substantially less efficient as lots of re-allocation is needed. It's better to guess the size, fill in and then rbind on a lot more rows if the guess was too small.(from Brian Ripley)

1.7 Need yet another different kind of merge for data frames (11/08/2000)

Convert these two files

```
File1
C A T
File2
1 2 34 56
2 3 45 67
3 4 56 78
(from Stephen Arthur)
```

Into a new data frame that looks like:

```
C A T 1 2 34 56
C A T 2 3 45 67
C A T 3 4 56 78
```

This works:

```
repcbind <- function(x,y) {
  nx <- nrow(x)
  ny <- nrow(y)
  if (nx<ny)
    x <- apply(x,2,rep,length=ny)
  else if (ny<nx)
    y <- apply(y,2,rep,length=nx)
  cbind(x,y)
}
(from Ben Bolker)
```

1.8 Check if an object is NULL (06/04/2001)

NULL does not mean that something does not exist. It means that it exists, and it is nothing.

```
X <- NULL
```

This may be a way of clearing values assigned to X, or initializing a variable as "nothing."

Programs can check on whether X is null

```
if ( is.null(x)) { #then...}
```

If you load things, R does not warn you when they are not found, it records them as NULL. You have the responsibility of checking them. Use

```
is.null(list$component)
```

to check a thing named component in a thing named list.

Accessing non-existent dataframe columns with "[" does give an error, so you could do that instead.

```
>data(trees)
>trees$aardvark
NULL
>trees[,"aardvark"]
```

Error in [.data.frame(trees, , "aardvark") : subscript out of bounds (from Thomas Lumley)

1.9 Generate random numbers (12/02/2012)

You want randomly drawn integers? Use Sample, like so:

```
# If you mean sampling without replacement:
>sample(1:10,3,replace=FALSE)
#If you mean with replacement:
>sample(1:10,3,replace=TRUE)
(from Bill Simpson)
```

Included with R are many univariate distributions, for example the Gaussian normal, Gamma, Binomial, Poisson, and so forth. Run

```
?runif
?rnorm
?rgamma
?rpois
```

You will see a distribution's functions are a base name like "norm" with prefix letters "r", "d", "p", "q".

- rnorm: draw pseudo random numbers from a normal
- dnorm: the density value for a given value of a variable
- pnorm: the cumulative probability density value for a given value
- qnorm: the quantile function: given a probability, what is the corresponding value of the variable?

I made a long-ish lecture about this in my R workshop (`http://pj.freefaculty.org/guides/Rcourse/rRandomVariables`)

Multivariate distributions are not (yet) in the base of R, but they are in several packages, such as MASS and mvtnorm. Note, when you use these, it is necessary to specify a mean vector and a covariance matrix among the variables. Brian Ripley gave this example: with mvrnorm in package MASS (part of the VR bundle),

```
mvrnorm(2, c(0,0), matrix(c(0.25, 0.20, 0.20, 0.25), 2,2))
```

If you don't want to use a contributed package to draw multivariate observations, you can approximate some using the univariate distributions in R itself. Peter Dalgaard observed "a less general solution for this particular case would be"

```
rnorm(1,sd=sqrt(0.20)) + rnorm(2,sd=sqrt(0.05))
```

1.10 Generate random numbers with a fixed mean/variance (06/09/2000)

If you generate random numbers with a given tool, you don't get a sample with the exact mean you specify. A generator with a mean of 0 will create samples with varying means, right?

I don't know why anybody wants a sample with a mean that is exactly 0, but you can draw a sample and then transform it to force the mean however you like. Take a 2 step approach:

```
R> x <- rnorm(100, mean = 5, sd = 2)
R> x <- (x - mean(x))/sqrt(var(x))
R> mean(x)
[1] 1.385177e-16
R> var(x)
[1] 1
```

and now create your sample with mean 5 and sd 2:

```
R> x <- x*2 + 5
R> mean(x)
[1] 5
R> var(x)
[1] 4
(from Torsten.Hothorn)
```

1.11 Use rep to manufacture a weighted data set (30/01/2001)

```
> x <- c(10,40,50,100)   # income vector for instance
> w <- c(2,1,3,2)        # the weight for each observation in x with the same
> rep(x,w)
[1]  10  10  40  50  50  50 100 100
(from P. Malewski)
```

That expands a single variable, but we can expand all of the columns in a dataset one at a time to represent weighted data

Thomas Lumley provided an example: Most of the functions that have weights have frequency weights rather than probability weights: that is, setting a weight equal to 2 has exactly the same effect as replicating the observation.

```
expanded.data<-as.data.frame(lapply(compressed.data,
                    function(x) rep(x,compressed.data$weights)))
```

1.12 Convert contingency table to data frame (06/09/2000)

Given a 8 dimensional crosstab, you want a data frame with 8 factors and 1 column for frequencies of the cells in the table.

R1.2 introduces a function as.data.frame.table() to handle this.

This can also be done manually. Here's a function (it's a simple wrapper around expand.grid):

```
dfify <- function(arr, value.name = "value", dn.names = names(dimnames(arr))) {
 Version <- "$Id: dfify.sfun,v 1.1 1995/10/09 16:06:12 d3a061 Exp $"
 dn <- dimnames(arr <- as.array(arr))
 if(is.null(dn))
    stop("Can't data-frame-ify an array without dimnames")
    names(dn) <- dn.names
    ans <- cbind(expand.grid(dn), as.vector(arr))
    names(ans)[ncol(ans)] <- value.name
    ans
}
```

The name is short for "data-frame-i-fy".

For your example, assuming your multi-way array has proper dimnames, you'd just do:

```
my.data.frame <- dfify(my.array, value.name=``frequency'')
```

(from Todd Taylor)

1.13 Write: data in text file (31/12/2001)

Say I have a command that produced a 28 x 28 data matrix. How can I output the matrix into a txt file (rather than copy/paste the matrix)?

```
write.table(mat, file="filename.txt")
```

Note MASS library has a function write.matrix which is faster if you need to write a numerical matrix, not a data frame. Good for big jobs.

2 Working with data frames: Recoding, selecting, aggregating

2.1 Add variables to a data frame (or list) (02/06/2003)

If dat is a data frame, the column x1 can be added to dat in (at least 4) methods, dat$x1, dat[, "x1"], dat["x1], or dat[["x1"]]. Observe

```
> dat <- data.frame(a=c(1,2,3))
> dat[ , "x1"] <- c(12, 23, 44)
> dat[ "x2"] <- c(12, 23, 44)
> dat[[ "x3"]] <- c(12, 23, 44)
> dat
  a x1 x2 x3
1 1 12 12 12
2 2 23 23 23
3 3 44 44 44
```

There are many other ways, including cbind().

Often I plan to calculate variable names within a program, as well as the values of the variables. I think of this as "generating new column names on the fly." In r-help, I asked "I keep finding myself in a situation where I want to calculate a variable name and then use it on the left hand side of an assignment." To me, this was a difficult problem.

Brian Ripley pointed toward one way to add the variable to a data frame:

```
iteration <- 9
newname <- paste("run", iteration, sep="")
mydf[newname] <- aColumn
## or, in one step:
mydf[paste("run",iteration,sep="")] <- aColumn
## for a list, same idea works, use double brackets
myList[[paste("run",iteration,sep="")]] <- aColumn
```

And Thomas Lumley added: " If you wanted to do something of this sort for which the above didn't work you could also learn about substitute():

```
eval(substitute(myList$newColumn<-aColumn),
     list(newColumn=as.name(varName)))
```

2.2 Create variable names "on the fly" (10/04/2001)

The previous showed how to add a column to a data frame on the fly. What if you just want to calculate a name for a variable that is not in a data frame. The assign function can do that. Try this to create an object (a variable) named zzz equal to 32.

```
> assign("zzz", 32)
> zzz
[1] 32
```

In that case,I specify zzz, but we can use a function to create the variable name. Suppose you want a random variable name. Every time you run this, you get a new variable starting with "a".

```
assign(paste("a", round(rnorm(1, 50,12), 2), sep=""), 324)
```

I got a44.05:

```
> a44.05
[1] 324
```

2.3 Recode one column, output values into another column (12/05/2003)

Please read the documentation for transform() and replace() and also learn how to use the magic of R vectors.

The transform() function works only for data frames. Suppose a data frame is called "mdf" and you want to add a new variable "newV" that is a function of var1 and var2:

```
mdf <- transform(mdf, newV=log(var1) + var2))
```

I'm inclined to take the easy road when I can. Proper use of indexes in R will help a lot, especially for recoding discrete valued variables. Some cases are particularly simple because of the way arrays are processed.

Suppose you create a variable, and then want to reset some values to missing. Go like this:

```
x <- rnorm(10000)  x[x > 1.5] <- NA
```

And if you don't want to replace the original variable, create a new one first (xnew <- x) and then do that same thing to xnew.

You can put other variables inside the brackets, so if you want x to equal 234 if y equals 1, then

```
x[y==1] <- 234
```

Suppose you have v1, and you want to add another variable v2 so that there is a translation. If v1=1, you want v2=4. If v1=2, you want v2=4. If v1=3, you want v2=5. This reminds me of the old days using SPSS and SAS. I think it is clearest to do:

```
> v1 <- c(1,2,3)# now initialize v2> v2 <- rep( -9, length(v1))  # now recode v2>
    v2[v1==1] <- 4> v2[v1==2]<-4> v2[v1==3]<-5> v2[1] 4 4 5
```

Note that R's "ifelse" command can work too:

```
x<-ifelse(x>1.5,NA,x)
```

One user recently asked how to take data like a vector of names and convert it to numbers, and 2 good solutions appeared:

```
y  <- c( "OLDa", "ALL",  "OLDc", "OLDa", "OLDb", "NEW", "OLDb", "OLDa", "ALL","
    ...")> el <- c("OLDa", "OLDb", "OLDc", "NEW", "ALL")> match(y,el)[1]  1  5  3
     1  2  4  2  1  5 NA
```

or

```
> f <- factor(x,levels=c("OLDa", "OLDb", "OLDc", "NEW", "ALL") )> as.integer(f)>
     [1] 1 5 3 1 2 4 2 1 5
```

I asked Mark Myatt for more examples:

For example, suppose I get a bunch of variables coded on a scale

```
1 = no   6 = yes   8 = tied   9 = missing  10 = not applicable.
```

Recode that into a new variable name with 0=no, 1=yes, and all else NA.

It seems like the replace() function would do it for single values but you end up with empty levels in factors but that can be fixed by re-factoring the variable. Here is a basic recode() function:

```
recode <- function(var, old, new){
  x <- replace(var, var==old, new)
  if(is.factor(x)) factor(x)
  else x
}
```

For the above example:

```
test <- c(1,1,2,1,1,8,1,2,1,10,1,8,2,1,9,1,2,9,10,1)
testtest <- recode(test, 1, 0)
test <- recode(test, 2, 1)
test <- recode(test, 8, NA)
test <- recode(test, 9, NA)
test <- recode(test, 10, NA)test
```

Although it is probably easier to use replace():

```
test <- c(1,1,2,1,1,8,1,2,1,10,1,8,2,1,9,1,2,9,10,1)
testtest <- replace(test, test==8 | test==9 | test==10, NA)
test <- replace(test, test==1, 0)
test <- replace(test, test==2, 1)test
```

I suppose a better function would take from and to lists as arguments:

```
recode <- function(var, from, to){
  x <- as.vector(var)
  for (i in 1:length(from)){
    x <- replace(x, x==from[i], to[i])
  }
 if(is.factor(var)) factor(x)
 else x
}
```

For the example:

```
test <- c(1,1,2,1,1,8,1,2,1,10,1,8,2,1,9,1,2,9,10,1)
testtest <- recode(test, c(1,2,8:10), c(0,1))
test
```

and it still works with single values.

Suppose somebody gives me ascale from 1 to 100, and I want to collapse it into 10 groups, how do I go about it?

Mark says: Use cut() for this. This cuts into 10 groups:

```
test <- trunc(runif(1000,1,100))
groups <-cut(test,seq(0,100,10))
table(test, groups)
```

To get ten groups without knowing the minimum and maximum value you can use pretty():

```
groups <- cut(test, pretty(test,10))
table(test, groups)
```

You can specify the cut-points:

```
groups <- cut(test,c(0,20,40,60,80,100))
table(test, groups)
```

And they don't need to be even groups:

```
groups <- cut(test,c(0,30,50,75,100))
table(test, groups)
```

Mark added, "I think I will add this sort of thing to the REX pack."

2003–12–01, someone asked how to convert a vector of numbers to characters, such as

```
if x[i] < 250 then col[i] = ``red''
 else if x[i] < 500 then col[i] = ``blue''
```

and so forth. Many interesing answers appeared in R-help. A big long nested ifelse would work, as in:

```
x.col <- ifelse(x < 250, "red",
   ifelse(x<500, "blue", ifelse(x<750, "green", "black")))
```

There were some nice suggestions to use cut, such as Gabor Grothendeick's advice:

The following results in a character vector:

```
> colours <- c("red", "blue", "green","back")
> colours[cut(x, c(-Inf,250,500,700,Inf),right=F,lab=FALSE)]
```

While this generates a factor variable:

```
> colours <- c("red", "blue", "green","black")
> cut(x, c(-Inf,250,500,700,Inf),right=F,lab=colours)
```

2.4 Create indicator (dummy) variables (20/06/2001)

2 examples:

c is a column, you want dummy variable, one for each valid value. First, make it a factor, then use model.matrix():

```
> x <- c(2,2,5,3,6,5,NA)
> xf <- factor(x,levels=2:6)
> model.matrix( xf-1 )
  xf2 xf3 xf4 xf5 xf6
1   1   0   0   0   0
2   1   0   0   0   0
3   0   0   0   1   0
4   0   1   0   0   0
5   0   0   0   0   1
6   0   0   0   1   0
attr(,"assign")
[1] 1 1 1 1 1
```

(from Peter Dalgaard)

Question: I have a variable with 5 categories and I want to create dummy variables for each category.

Answer: Use row indexing or model.matrix.

```
ff <- factor(sample(letters[1:5], 25, replace=TRUE))
diag(nlevels(ff))[ff,]
#or
model.matrix(~ff - 1)
(from Brian D. Ripley)
```

2.5 Create lagged values of variables for time series regression (05/22/2012)

Peter Dalgaard explained, "the simple way is to create a new variable which shifts the response, i.e.

```
yshft <- c(y[-1], NA) # pad with missing
summary(lm(yshft ~ x + y))
```

Alternatively, lag the regressors:

```
N <- length(x)
xlag <- c(NA, x[1:(N-1)])
ylag <- c(NA, y[1:(N-1)])
summary(lm(y ~ xlag + ylag))
```

Dave Armstrong (personal communication, 2012/5/21) brought to my attention the following problem in "cross sectional time series" data. Simply inserting an NA will lead to disaster because we need to insert a lag "within" each unit. There is also a bad problem when the time points observed for the sub units are not all the same. He suggests the following

```
dat <- data.frame(
  ccode = c(1,1,1,1,2,2,2,2,3,3,3,3),
  year = c(1980, 1982, 1983, 1984, 1981:1984, 1980:1982, 1984),
  x = seq(2,24, by=2))

dat$obs <- 1:nrow(dat)
dat$lagobs <- match(paste(dat$ccode, dat$year-1, sep="."),
paste(dat$ccode, dat$year, sep="."))
dat$lag_x <- dat$x[dat$lagobs]
```

Run this example, be surprised, then email me if you find a better way. I haven't. This seems like a difficult problem to me and if I had to do it very often, I am pretty sure I'd have to re-think what it means to "lag" when there are years missing in the data. Perhaps this is a rare occasion where interpolation might be called for.

2.6 How to drop factor levels for datasets that don't have observations with those values? (08/01/2002)

The best way to drop levels, BTW, is

```
problem.factor <- problem.factor[ ,drop=TRUE]
(from Brian D. Ripley)
```

That has the same effect as running the pre-existing "problem.factor" through the function factor:

```
problem.factor <- factor(problem.factor)
```

2.7 Select/subset observations out of a dataframe (08/02/2012)

If you just want particular named or numbered rows or columns, of course, that's easy. Take columns x1, x2, and x3.

```
datSubset1 <- dat[ , c("x1","x2","x3")]
```

If those happen to be columns 44, 92, and 93 in a data frame,

```
datSubset1 <- dat[ , c(44, 92, 93)]
```

Usually, we want observations that are conditional.

Want to take observations for which variable Y is greater than A and less or equal than B:

```
X[Y > A & Y ≤ B ]
```

Suppose you want observations with c=1 in df1. This makes a new data frame.

```
df2 <- df1[df1$c==1,]
```

and note that indexing is pretty central to using S (the language), so it is worth learning all the ways to use it. (from Brian Ripley)

Or use "match" select values from the column "d" by taking the ones that match the values of another column, as in

```
> d <- t(array(1:20,dim=c(2,10)))
> i <- c(13,5,19)
> d[match(i,d[,1]), 2]
[1] 14  6 20
(from Peter Wolf)
```

Till Baumgaertel wanted to select observations for men over age 40, and sex was coded either m or M. Here are two working commands:

```
# 1.)
maleOver40 <- subset(d, sex %in% c("m","M") & age > 40)
# 2.)
maleOver40 <- d[(d$sex=="m" | d$sex=="M") & d$age >40,]
```

To decipher that, do ?match and ?"%in" to find out about the %in% operator.

If you want to grab the rows for which the variable "subject" is 15 or 19, try:

```
df1$subject %in% c(19,15)
```

to get a True/False indication for each row in the data frame, and you can then use that output to pick the rows you want:

```
indicator <- df1$subject %in% c(19,15)
df1[indicator,]
```

How to deal with values that are already marked as missing? If you want to omit all rows for which one or more column is NA (missing):

```
x2 <- na.omit(x)
```

produces a copy of the data frame x with all rows that contain missing data removed. The function na.exclude could be used also. For more information on missings, check help : ?na.exclude.

For exclusion of missing, Peter Dalgaard likes

```
subset(x,complete.cases(x)) or x[complete.cases(x),]
```

adding "is.na(x) is preferable to x !="NA"

2.8 Delete first observation for each element in a cluster of observations (11/08/2000)

Given data like:

```
1  ABK     19910711     11.1867461      0.0000000       108
2  ABK     19910712     11.5298979     11.1867461       111
6 CSCO   19910102      0.1553819      0.0000000        106
7 CSCO   19910103      0.1527778      0.1458333        166
```

remove the first observation for each value of the "sym" variable (the one coded ABK,CSCO, etc). . If you just need to remove rows 1, 6, and 13, do:

```
newhilodata <- hilodata[-c(1,6,13),]
```

To solve the more general problem of omitting the first in each group, assuming "sym" is a factor, try something like

```
newhilodata <- subset(hilodata, diff(c(0,as.integer(sym))) != 0)
```

(actually, the as.integer is unnecessary because the c() will unclass the factor automagically)

(from Peter Dalgaard)

Alternatively, you could use the match function because it returns the first match. Suppose jm is the data set. Then:

```
> match(unique(jm$sym), jm$sym)
 [1]  1  6 13
> jm <- jm[ -match(unique(jm$sym), jm$sym), ]
```

(from Douglas Bates)

As Robert pointed out to me privately: duplicated() does the trick

```
subset(hilodata, duplicated(sym))
```

has got to be the simplest variant.

2.9 Select a random sample of data (11/08/2000)

```
sample(N, n, replace=FALSE)
```

and

```
seq(N)[rank(runif(N)) ≤ n]
```

is another general solution. (from Brian D. Ripley)

2.10 Selecting Variables for Models: Don't forget the subset function (15/08/2000)

You can manage data directly by deleting lines or so forth, but subset() can be used to achieve the same effect without editing the data at all. Do ?select to find out more. Subset is also an option in many statistical functions like lm.

Peter Dalgaard gave this example, using the "builtin" dataset airquality.

```
data(airquality)
names(airquality)
lm(Ozone~., data=subset(airquality,select=Ozone:Month))
lm(Ozone~., data=subset(airquality,select=c(Ozone:Wind,Month)))
lm(Ozone~.-Temp, data=subset(airquality,select=Ozone:Month))
```

The "." on the RHS of lm means "all variables" and the subset command on the rhs picks out different variables from the dataset. "x1:x2" means variables between x1 and x2, inclusive.

2.11 Process all numeric variables, ignore character variables? (11/02/2012)

This was a lifesaver for me. I ran simulations in C and there were some numbers and some character variables. The output went into "lastline.txt" and I wanted to import the data and leave some in character format, and then calculate summary indicators for the rest.

```
data <- read.table(``lastline.txt'',header=TRUE, as.is = TRUE)
indices <- 1:dim(data)[2]
indices <- na.omit(ifelse(indices*sapply(data,is.numeric),indices,NA))
mean <- sapply(data[,indices],mean)
sd <- sapply(data[,indices],sd)
```

If you just want a new dataframe with only the numeric variables, do:

```
sapply(dataframe, is.numeric)
# or
which(sapply(data.frame, is.numeric))
```

(Thomas Lumley)

2.12 Sorting by more than one variable (06/09/2000)

Can someone tell me how can I sort a list that contains duplicates (name) but keeping the duplicates together when sorting the values.

```
name  M
1234  8
1234  8.3
4321  9
4321  8.1
```

I also would like to set a cut-off, so that anything below a certain values will not be sorted.(from Kong, Chuang Fong)

I take it that the cutoff is on the value of M. OK, suppose it is the value of 'coff'.

```
sort.ind <- order(name, pmax(coff, M))  # sorting index
name <- name[sort.ind]
M <- M[sort.ind]
```

Notice how using pmax() for "parallel maximum" you can implement the cutoff by raising all values below the mark up to the mark thus putting them all into the same bin as far as sorting is concerned.

If your two variables are in a data frame you can combine the last two steps into one, of course.

```
sort.ind <- order(dat$name, pmax(coff, dat$M))
dat <- dat[sort.ind, ]
```

In fact it's not long before you are doing it all in one step:

```
dat <- dat[order(dat$name, pmax(coff, dat$M)), ]
(from Bill Venables)
```

I want the ability to sort a data frame lexicographically according to several variables
Here's how:

```
spsheet[order(name,age,zip),]
(from Peter Dalgaard)
```

2.13 Rank within subgroups defined by a factor (06/09/2000)

Read the help for by()

```
> by(x[2], x$group, rank)
x$group: A
[1] 4.0 1.5 1.5 3.0
------------------------------------------------------------
x$group: B
[1] 3 2 1

> c(by(x[2], x$group, rank),  recursive=TRUE)
 A1  A2  A3  A4  B1  B2  B3
4.0 1.5 1.5 3.0 3.0 2.0 1.0
```

(from Brian D. Ripley)

2.14 Work with missing values (na.omit, is.na, etc) (15/01/2012)

NA is a "symbol", not just a pair of letters. Values for any type of variable, numeric or character, may be set to NA. Thus, in a sense, management of NA values is just like any other recoding exercise. Find the NAs and change them, or find suspicious values and set them to NA. Matt Wiener (2005–01–05) offered this example, which inserts some NA's, and then converts them all to 0.

```
> temp1 <- matrix(runif(25), 5, 5)
> temp1[temp1 < 0.1] <- NA
> temp1[is.na(temp1)] <- 0
```

When routines encounter NA values, special procedures may be called for.

For example, the mean function will return NA when it encounters any NA values. That's a cold slap in the face to most of us. To avoid that, run mean(x, na.rm=TRUE), so the NA's are automatically ignored. Many functions have that same default (check max, min).

Other functions, like lm, will default to omit missings, unless the user has put other settings in place. The environment options() has settings that affect the way missings are handled.

Suppose you are a cautious person. If you want lm to fail when it encounters an NA, then the lm argument na.action can be set to na.fail. Before that, however, it will be necessary to change or omit missings. To manually omit missings, then sure missings are omitted by telling lm to fail if it finds NAs.

```
thisdf <- na.omit(insure[,c(1, 19:39)])
body.m <- lm(BI.PPrem ~ ., data = thisdf, na.action = na.fail)
```

The function is.na() returns a TRUE for missings, and that can be used to spot and change them.

The table function defaults to exclude NA from the reported categories. To force it to report NA as a valid value, add the option exclude=NULL in the table command.

2.15 Aggregate values, one for each line (16/08/2000)

Question: I want to read values from a text file - 200 lines, 32 floats per line - and calculate a mean for each of the 32 values, so I would end up with an "x" vector of 1–200 and a "y" vector of the 200 means.

Peter Dalgaard says do this:

```
y <- apply(as.matrix(read.table(myfile)), 1, mean)
x <- seq(along=y)
```

(possibly adding a couple of options to read.table, depending on the file format)

2.16 Create new data frame to hold aggregate values for each factor (11/08/2000)

How about this:

```
Z<-aggregate(X, f, sum)
```

(assumes all X variables can be summed)

Or: [If] X contains also factors. I have to select variables for which summary statistics have to be computed. So I used:

```
Z <- data.frame(f=levels(f),x1=as.vector(tapply(x1,f,sum)))
(from Wolfgang Koller)
```

2.17 Selectively sum columns in a data frame (15/01/2012)

Given

```
10 20  23 44 33
10 20  33 23 67
```

and you want

```
10 20  56 67 100
```

try this, where it assumes that "data.dat" has two columns std and cf that we do not want to sum:

```
dat<-read.table("data.dat",header=TRUE)
aggregate(dat[,-(1:2)], by=list(std=dat$std, cf=dat$cf), sum)
```

note the first two columns are excluded by [,(1:2)] and the by option preserves those values in the output.

2.18 Rip digits out of real numbers one at a time (11/08/2000)

I want to "take out" the first decimal place of each output, plot them based on their appearance frequencies. Then take the second decimal place, do the same thing.

```
a<- log(1:1000)
d1<-floor(10*(a-floor(a)))   # first decimal
  par(mfrow=c(2,2))
  hist(d1,breaks=c(-1:9))
  table(d1)
d2<-floor(10*(10*a-floor(10*a))) # second decimal
  hist(d2,breaks=c(-1:9))
  table(d2)
(from Yudi Pawitan)
```

```
x <- 1:1000

ndig <- 6

 (ii <- as.integer(10^(ndig-1) * log(x)))[1:7]
 (ci <- formatC(ii, flag="0", wid= ndig))[1:7]
cm <- t(sapply(ci, function(cc) strsplit(cc,NULL)[[1]]))
cm [1:7,]

apply(cm, 2, table) #--> Nice tables

# The plots :
par(mfrow= c(3,2), lab = c(10,10,7))
for(i in 1:ndig)
    hist(as.integer(cm[,i]), breaks = -.5 + 0:10,
         main = paste("Distribution of  ", i,"-th digit"))
(from Martin Maechler)
```

2.19 Grab an item from each of several matrices in a List (14/08/2000)

Let Z denote the list of matrices. All matrices have the same order. Suppose you need to take element [1,2] from each.

```
lapply(Z, function(x) x[1,2])
```

should do this, giving a list. Use sapply if you want a vector. (Brian Ripley)

2.20 Get vector showing values in a dataset (10/04/2001)

```
xlevels <- sort(unique(x))
```

2.21 Calculate the value of a string representing an R command (13/08/2000)

In R, they use the term "expression" to refer to a command that is written down as a character string, but is not yet submitted to the processor. That has to be parsed, and then evaluated. Example:

```
String2Eval <- "A valid R statement"
eval(parse(text = String2Eval))
(from Mark Myatt)
```

Using eval is something like saying to R, "I'd consider typing this into the command line now, but I typed it before and it is a variable now, so couldn't you go get it and run it now?"

```
# Or
eval(parse(text="ls()"))
# Or
eval(parse(text = "x[3] <- 5"))
(from Peter Dalgaard)
```

Also check out substitute(), as.name() et al. for other methods of manipulating expressions and function calls

2.22 "Which" can grab the index values of cases satisfying a test (06/04/2001)

To analyze large vectors of data using boxplot to find outliers, try:

```
which(x==boxplot(x,range=1)$out)
```

2.23 Find unique lines in a matrix/data frame (31/12/2001)

Jason Liao writes:

"I have 10000 integer triplets stored in A[1:10000, 1:3]. I would like to find the unique triplets among the 10000 ones with possible duplications."

Peter Dalgaard answers, "As of 1.4.0 (!):

```
unique(as.data.frame(A))
```

3 Matrices and vector operations

3.1 Create a vector, append values (01/02/2012)

Some things in R are so startlingly simple and different from other programs that the experts have a difficult time understanding our misunderstanding.

Mathematically, I believe a vector is a column in which all elements are of the same type (e.g., real numbers). (1,2,3) is a vector of integers as well as a column in the data frame sense. If you put dissimilar-typed items into an R column, it creates a vector that reduces the type of all values to the lowest type. Example:

```
> a <- c("a", 1, 2, 3, "hello")
> is.vector(a)
[1] TRUE
> is.character(a)
[1] TRUE
> as.numeric(a)
[1] NA  1  2  3 NA
Warning message: NAs introduced by coercion
```

This shows that an easy way to create a vector is with the concatenate function, which is used so commonly it is known only as "c". As long as all of your input is of the same type, you don't hit any snags. Note the is.vector() function will say this is a vector:

```
> v <- c(1, 2, 3)
> is.vector(v)
[1] TRUE
> is.numeric(v)
[1] TRUE
```

The worst case scenario is that it manufactures a list. The double brackets [[]] are the big signal that you really have a list:

```
> bb <- c(1,c,4)
> is.vector(bb)
[1] TRUE
> bb
[[1]]
[1] 1
[[2]]
.Primitive("c")
[[3]]
[1] 4
> is.character(bb)
[1] FALSE
> is.integer(bb)
[1] FALSE
> is.list(bb)
[1] TRUE
```

To make sure you get what you want, you can use the vector() function, and then tell it what type of elements you want in your vector. Please note it puts a "false" value in each position, not a missing.

```
> vector(mode="integer",10)
 [1] 0 0 0 0 0 0 0 0 0 0
> vector(mode="double",10)
 [1] 0 0 0 0 0 0 0 0 0 0
> vector(mode="logical",10)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

There are also methods to directly create the particular kind of vector you want (factor, character, integer, etc). For example, if you want a real-valued vector, there is a shortcut method numeric()

```
> x <- numeric(10)
> x
 [1] 0 0 0 0 0 0 0 0 0 0
```

If you want to add values onto the end of a vector, you have options!

If you want to add a single value onto the vector, try:

```
> r <- rnorm(1)
> v <- append(v, r)
```

The length() command can not only tell you how long a vector is, but it can be used to RESIZE it. This command says if the length of v has reached the value vlen, then expand the length by doubling it.

```
if (vlen==length(v)) length(v) <- 2*length(v)
```

These were mentioned in an email from Gabor Grothendieck to the list on 2005–01–04.

3.2 How to create an identity matrix? (16/08/2000)

```
diag(n)
```

Or, go the long way:

```
n<-c(5)
I <- matrix(0,nrow=n,ncol=n)
I[row(I)==col(I)] <- 1
(from E.D.Isaia)
```

3.3 Convert matrix "m" to one long vector (11/08/2000)

```
dim(m)<-NULL
```

or

```
c(m)
(from Peter Dalgaard )
```

3.4 Creating a peculiar sequence (1 2 3 4 1 2 3 1 2 1) (11/08/2000)

I don't know why this is useful, but it shows some row and matrix things:

I ended up using Brian Ripley's method, as I got it first and it worked, ie.

```
A <- matrix(1, n-1, n-1)
rA <- row(A)
rA[rA + col(A) <= n]
```

However, thanks to Andy Royle I have since discovered that there is a much more simple and sublime solution:

```
> n <- 5
> sequence((n-1):1)
[1] 1 2 3 4 1 2 3 1 2 1
(from Karen Kotschy)
```

3.5 Select every n'th item (14/08/2000)

extract every nth element from a very long vector, vec? We need to create an "index vector" to indicate which values we want to take. The seq function creates a sequence that can step from one value to another, by a certain increment.

```
seq(n, length(vec), by=n)
(Brian Ripley)
```

```
seq(1,11628,length=1000)
```

will give 1000 evenly spaced numbers from 1:11628 that you can then index with. (from Thomas Lumley)

My example. Use the index vector to take what we need:

```
> vec <- rnorm(1999)
> newvec <- vec[ seq(1, length(vec), 200) ]
> newvec
 [1]  0.2685562  1.8336569  0.1371113  0.2204333 -1.2798172  0.3337282
 [7] -0.2366385  0.5060078  0.9680530  1.2725744
```

This shows the items in vec at indexes (1, 201, 401, . . . , 1801)

3.6 Find index of a value nearest to 1.5 in a vector (11/08/2000)

```
n <- 1000
x <- sort(rnorm(n))
x0 <- 1.5
dx <- abs(x-x0)
which(dx==min(dx))
```

(from Jan Schelling)

```
which(abs(x - 1.5)==min(abs(x - 1.5)))
```

(from Uwe Ligges)

3.7 Find index of nonzero items in vector (18/06/2001)

```
which(x!=0)
(from Uwe Ligges)
```

```
rfind <- function(x) seq(along=x)[x != 0]
(from Brian D. Ripley)
```

Concerning speed and memory efficiency I find

```
as.logical(x)
```

is better than

```
x!=0
```

and

```
seq(along=x)[as.logical(x)]
```

is better than

```
which(as.logical(x))
```

thus

```
which(x!=0)
```

is shortest and

```
rfind <- function(x)seq(along=x)[as.logical(x)]
```

seems to be computationally most efficient

(from Jens Oehlschlagel-Akiyoshi)

3.8 Find index of missing values (15/08/2000)

Suppose the vector "Pes" has 600 observations. Don't do this:

```
(1:600)[is.na(Pes)]
```

The 'approved' method is

```
seq(along=Pes)[is.na(Pes)]
```

In this case it does not matter as the subscript is of length 0, but it has floored enough library/-package writers to be worth thinking about.

(from Brian Ripley)

However, the solution I gave

```
which(is.na(Pes))
```

is the one I still really recommend; it does deal with 0-length objects, and it keeps names when there are some, and it has an 'arr.ind = FALSE' argument to return array indices instead of vector indices when so desired. (from Martin Maechler)

3.9 Find index of largest item in vector (16/08/2000)

```
A[which(A==max(A,na.rm=TRUE))]
```

3.10 Replace values in a matrix (22/11/2000)

```
> tmat <- matrix(rep(0,3*3),ncol=3)
> tmat
[,1] [,2] [,3]
[1,] 0 0 0
[2,] 0 0 0
[3,] 0 0 0
> tmat[tmat==0] <-1
> tmat
[,1] [,2] [,3]
[1,] 1 1 1
[2,] 1 1 1
[3,] 1 1 1
(from Jan Goebel)
```

If la is a data frame, you have to coerce the data into matrix form, with:

```
la <- as.matrix(la)
la[la==0] <- 1
```

Try this:

```
la <- ifelse(la==0, 1, la)
```

(from Martyn Plummer)

3.11 Delete particular rows from matrix (06/04/2001)

```
> x <- matrix(1:10,,2)
> x[x[,1] %in% c(2,3),]
> x[!x[,1] %in% c(2,3),]
```

(from Peter Malewski)

```
mat[!(mat$first %in% 713:715),]
```

(from Peter Dalgaard)

3.12 Count number of items meeting a criterion (01/05/2005)

Apply "length()" to results of which() described in previous question, as in

```
length(which(v<7))
# or
sum( v<7,na.rm=TRUE)
```

If you apply sum() to a matrix, it will scan over all columns. To focus on a particular column, use subscripts like sum(v[,1]>0) or such. If you want separate counts for columns, there's a method colSums() that will count separately for each column.

3.13 Compute partial correlation coefficients from correlation matrix (08/12/2000)

I need to compute partial correlation coefficients between multiple variables (correlation between two paired samples with the "effects of all other variables partialled out")? (from Kaspar Pflugshaupt)

Actually, this is quite straightforward. Suppose that R is the correlation matrix among the variables. Then,

```
Rinv<-solve(R)
        D<-diag(1/sqrt(diag(Rinv)))
        P<- -D %*% Rinv %*% D
```

The off-diagonal elements of P are then the partial correlations between each pair of variables "partialed" for the others. (Why one would want to do this is another question.)

(from John Fox).

In general you invert the variance-covariance matrix and then rescale it so the diagonal is one. The off-diagonal elements are the negative partial correlation coefficients given all other variables.

```
pcor2 <- function(x){
  conc <- solve(var(x))
  resid.sd <- 1/sqrt(diag(conc))
  pcc <- -sweep(sweep(conc, 1, resid.sd, "*"), 2, resid.sd, "*")
  pcc
}

pcor2(cbind(x1,x2,x3))
```

see J. Whittaker's book "Graphical models in applied multivariate statistics" (from Martyn Plummer)

This is the version I'm using now, together with a test for significance of each coefficient (H0: coeff=0):

```
f.parcor <-
function (x, test = F, p = 0.05)
{
    nvar <- ncol(x)
    ndata <- nrow(x)
    conc <- solve(cor(x))
    resid.sd <- 1/sqrt(diag(conc))
    pcc <- -sweep(sweep(conc, 1, resid.sd, "*"), 2, resid.sd,
        "*")
    colnames(pcc) <- rownames(pcc) <- colnames(x)
    if (test) {
        t.df <- ndata - nvar
        t <- pcc/sqrt((1 - pcc^2)/t.df)
        pcc <- list(coefs = pcc, significant = t > qt(1 - (p/2),
            df = t.df))
    }
    return(pcc)
}
```

(from Kaspar Pflugshaupt)

3.14 Create a multidimensional matrix (R array) (20/06/2001)

Brian Ripley said:

```
my.array<-array(0,dim=c(10,5,6,8))
```

will give you a 4-dimensional 10 x 5 x 6 x 8 array.

Or

```
array.test <- array(1:64,c(4,4,4))
array.test[1,1,1]
1
array.test[4,4,4]
64
```

3.15 Combine a lot of matrices (20/06/2001)

If you have a list of matrices, it may be tempting to take them one by one and use "rbind" to stack them all together. Don't! It is really slow because of inefficient memory allocation. Better to do

```
do.call("rbind", listOfMatrices)
```

In the WorkingExamples folder, I wrote up a longish example of this, "stackListItems.R".

3.16 Create "neighbor" matrices according to specific logics (20/06/2001)

Want a matrix of 0s and 1s indicating whether a cell has a neighbor at a location:

```
N <- 3
x <- matrix(1:(N^2),nrow=N,ncol=N)
rowdiff <- function(y,z,mat)abs(row(mat)[y]-row(mat)[z])
coldiff <- function(y,z,mat)abs(col(mat)[y]-col(mat)[z])
rook.case <- function(y,z,mat){coldiff(y,z,mat)+rowdiff(y,z,mat)==1}
bishop.case <- function(y,z,mat){coldiff(y,z,mat)==1 & rowdiff(y,z,mat)==1}
queen.case <- function(y,z,mat){rook.case(y,z,mat) | bishop.case(y,z,mat)}
matrix(as.numeric(sapply(x,function(y)sapply(x,rook.case,y,mat=x))),ncol=N^2,nrow
    =N^2)
matrix(as.numeric(sapply(x,function(y)sapply(x,bishop.case,y,mat=x))),ncol=N^2,
    nrow=N^2)
```

```
matrix(as.numeric(sapply(x,function(y)sapply(x,queen.case,y,mat=x))),ncol=N^2,
    nrow=N^2)
```

(from Ben Bolker)

3.17 "Matching" two columns of numbers by a "key" variable (20/06/2001)

The question was:

I have a matrix of predictions from an proportional odds model (using the polr function in MASS), so the columns are the probabilities of the responses, and the rows are the data points. I have another column with the observed responses, and I want to extract the probabilities for the observed responses.

As a toy example, if I have

```
> x <- matrix(c(1,2,3,4,5,6),2,3)
> y <- c(1,3)
```

and I want to extract the numbers in x[1,1] and x[2,3] (the columns being indexed from y), what do I do?

Is

x[cbind(seq(along=y), y)]

what you had in mind? The key is definitely matrix indexing. (from Brian Ripley)

3.18 Create Upper or Lower Triangular matrix (06/08/2012)

The most direct route to do that is to create a matrix X, and then use the upper.tri() or lower.tri() functions to take out the parts that are needed.

To do it in one step is trickly. Whenever it comes up in r-help, there is a different answer. Suppose you want a matrix like

```
a^0   0     0
a^1   a^0   0
a^2   a^1   a^0
```

I don't know why you'd want that, but look at the differences among answers this elicited.

```
ma <- matrix(rep(c(a^(0:n), 0), n+1), nrow=n+1, ncol=n+1)
ma[upper.tri(ma)] <- 0
```

(from Uwe Ligges)

```
> n <- 3
> a <- 2
> out <- diag(n+1)
> out[lower.tri(out)] <-
a^apply(matrix(1:n,ncol=1),1,function(x)c(rep(0,x),1:(n-x+1)))[lower.tri(out)]
> out
```

(from Woodrow Setzer)

If you want an upper triangular matrix, this use of "ifelse" with outer looks great to me:

```
> fun <- function(x,y) { ifelse(y>x, x+y, 0) }
> fun(2,3)
[1] 5
> outer(1:4, 1:4, fun)
[,1] [,2] [,3] [,4]
[1,]    0    3    4    5
[2,]    0    0    5    6
[3,]    0    0    0    7
[4,]    0    0    0    0
```

If I had to do this, this method is most understandable to me. Here set a=0.2.

```
> a <- 0.2
> fun <- function(x, y) { ifelse(y<=x, a^(x+y-1), 0) }
> outer(1:4, 1:4, fun)
       [,1]    [,2]    [,3]     [,4]
[1,] 0.2000 0.00000 0.0e+00 0.00e+00
[2,] 0.0400 0.00800 0.0e+00 0.00e+00
[3,] 0.0080 0.00160 3.2e-04 0.00e+00
[4,] 0.0016 0.00032 6.4e-05 1.28e-05
```

During the Summer Stats Camp at KU in 2012, a user wanted to import a file full of numbers that were written by Mplus and then convert it to a lower triangular matrix. These need to end up in row order, like so:

```
1 0 0 0 0
2 3 0 0 0
4 5 6 0 0
7 8 9 10 0
11 12 13 14 15
```

The data is a raw text file called "tech3.out". The first solution I found used a function to create symmetric matrices from the package miscTools (by Arne Henningsen and Ott Toomet). This creates a full matrix, and then I "obliterate" the upper right side by "writing over" with 0's.

```
x <- scan("tech3.out")
library(miscTools)

dat1 <- symMatrix( x, nrow=90, byrow = TRUE)
upperNoDiag <- upper.tri(dat1, diag = FALSE)
dat1[upperNoDiag] <- 0
fix(dat1)
```

To find out why that works, just type "symMatrix". The relevant part of Arne's code is

```
if (byrow != upper) {
    result[upper.tri(result, diag = TRUE)] <- data
    result[lower.tri(result)] <- t(result)[lower.tri(result)]
}
```

Once we fiddle around with that idiom a bit, we see that the magic is in the way R's matrix receives a vector and "automagically" fills in the pieces. Because we want to fill the rows, it turns out we have to first fill the columns of an upper triangular matrix and then transpose the result.

```
x <- 1:6
xmat <- matrix(0, 3 ,3)
xmat[upper.tri(xmat, diag = TRUE)] <- x
result <- t(xmat)

> result
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    2    3    0
[3,]    4    5    6
```

This pre-supposes I've got the correct number of elements in x. The main advantage of symMatrix for us is that it checks the size of the vector against the size of the matrix and reports an error if they don't match up properly.

3.19 Calculate inverse of X (12/02/2012)

solve(A) yields the inverse of A.

Caution: Be sure you really want an inverse. To estimate regression coefficients, the recommended method is not to solve $(X'X)^{-1}X'y$, but rather use a QR decomposition. There's a nice discussion of that in *Generalized Additive Models* by Simon Wood.

3.20 Interesting use of Matrix Indices (20/06/2001)

Mehdi Ghafariyan said "I have two vectors A=c(5,2,2,3,3,2) and
B=c(2,3,4,5,6,1,3,2,4,3,1,5,1,4,6,1,4) and I want to make the following
matrix using the information I have from the above vectors.

```
0 1 1 1 1 1
1 0 1 0 0 0
0 1 0 1 0 0
1 0 1 0 1 0
1 0 0 1 0 1
1 0 0 1 0 0
```

so the first vector says that I have 6 elements therefore I have to make a 6 by 6 matrix and then I have to read 5 elements from the second vector , and put 1 in [1,j] where j=2,3,4,5,6 and put zero elsewhere(i.e.˜in [1,1]) and so on. Any idea how this can be done in R ?

Use matrix indices:

```
a<-c(5,2,2,3,3,2)
b<-c(2,3,4,5,6,1,3,2,4,3,1,5,1,4,6,1,4)
n<-length(a)
M<-matrix(0,n,n)
M[cbind(rep(1:n,a),b)]<-1
(from Peter Dalgaard )
```

3.21 Eigenvalues example (20/06/2001)

Tapio Nummi asked about double-checking results of spectral decomposition

In what follows, m is this matrix:

```
0.4015427  0.08903581 -0.2304132
0.08903581 1.60844812  0.9061157
-0.23041322 0.9061157  2.9692562
```

Brian Ripley posted:

```
> sm <- eigen(m, sym=TRUE)
> sm
$values
[1] 3.4311626 1.1970027 0.3510817

$vectors
            [,1]        [,2]       [,3]
[1,] -0.05508142 -0.2204659  0.9738382
[2,]  0.44231784 -0.8797867 -0.1741557
[3,]  0.89516533  0.4211533  0.1459759

> V <- sm$vectors
> t(V) %*% V
              [,1]          [,2]          [,3]
[1,]  1.000000e+00 -1.665335e-16 -5.551115e-17
[2,] -1.665335e-16  1.000000e+00  2.428613e-16
[3,] -5.551115e-17  2.428613e-16  1.000000e+00

> V %*% diag(sm$values) %*% t(V)
            [,1]        [,2]       [,3]
```

```
[1,]  0.40154270 0.08903581 -0.2304132
[2,]  0.08903581 1.60844812  0.9061157
[3,] -0.23041320 0.90611570  2.9692562
```

4 Applying functions, tapply, etc

4.1 Return multiple values from a function (12/02/2012)

Please note. The return function is NOT required at the end of a function.

One can return a collection of objects in a list, as in

```
fun <- function(z){
##whatever...
   list(x1, x2, x3 ,x4)
}
```

One can also return a single object from a function, but include additional information as attributes of the single thing. For example,

```
fun <- function(x,y,z){
  m1 <- lm(y ~ x + z)
  attr(m1, "friend") <- "joe"
}
```

I don't want to add the character variable "joe" to any regression models, but it is nice to know I could do that.

4.2 Grab "p" values out of a list of significance tests (22/08/2000)

Suppose chisq1M11 is a list of htest objects, and you want a vector of p values. Kjetil Kjernsmo observed this works:

```
> apply(cbind(1:1000), 1, function(i) chisq1M11[[i]]$p.value)
```

And Peter Dalgaard showed a more elegant approach:

```
sapply(chisq1M11,function(x)x$p.value)
```

In each of these, a simple R function called "function" is created and applied to each item

4.3 ifelse usage (12/02/2012)

If y is 0, leave it alone. Otherwise, set y equal to y*log(y)

```
ifelse(y==0, 0, y*log(y))
(from Ben Bolker)
```

4.4 Apply to create matrix of probabilities, one for each "cell" (14/08/2000)

Suppose you want to calculate the gamma density for various values of "scale" and "shape". So you create vectors "scales" and "shapes", then create a grid if them with expand.grid, then write a function, then apply it (this example courtesy of Jim Robison-Cox) :

```
gridS <- expand.grid(scales, shapes)
survLilel <- function(ss) sum(dgamma(Survival,ss[1],ss[2]))
Likel <- apply(gridS,1,survLilel)
```

Actually, I would use

```
sc <- 8:12; sh <- 7:12
args <- expand.grid(scale=sc, shape=sh)
matrix(apply(args, 1, function(x) sum(dgamma(Survival, scale=x[1],
shape=x[2], log=TRUE))), length(sc), dimnames=list(scale=sc, shape=sh))
(from Brian Ripley)
```

4.5 Outer. (15/08/2000)

outer(x,y,f) does just one call to f with arguments created by stacking x and y together in the right way, so f has to be vectorised. (from Thomas Lumley)

4.6 Check if something is a formula/function (11/08/2000)

Formulae have class "formula", so inherits(obj, "formula") looks best. (from Prof Brian D Ripley)

And if you want to ensure that it is ~X+Z+W rather than

Y ~ X+Z+W you can use

```
inherits(obj,``formula'') && (length(obj)==2)
```

(from Thomas Lumley)

4.7 Optimize with a vector of variables (11/08/2000)

The function being optimized has to be a function of a single argument.

If alf and bet are both scalars you can combine them into a vector and use

```
opt.func <- function(arg)
  t(Y-(X[,1] * arg[1] + X[,2] * arg[2])^delta) %*% covariance.matrix.inverse %*%
            (Y-(X[,1] * arg[1] + X[,2] * arg[2])^delta)
(from Douglas Bates)
```

4.8 slice.index, like in S+ (14/08/2000)

```
slice.index <-function(x,i){
k <-dim(x)[i]
sweep(x,i,1:k,function(x,y) y)
}
(from Peter Dalgaard)
```

5 Graphing

5.1 Adjust features with par before graphing (18/06/2001)

par() is a multipurpose command to affect qualities of graphs. par(mfrow=c(3,2)), creates a figure that has a 3x2 matrix of plots, or par(mar=c(10, 5, 4, 2)), adjust margins. A usage example:

```
par(mfrow=c(2,1))
par(mar=c(10, 5, 4, 2))
x <- 1:10
plot(x)
box("figure", lty="dashed")
par(mar=c(5, 5, 10, 2))
plot(x)
box("figure", lty="dashed")
```

5.2 Save graph output (03/21/2014)

There are two ways to save and/or print graphs. You can create the graphs on screen, and then copy them into a file, or you can write them directly into a file. This is explained in greater depth in my R lecture notes, plot-2 (`http://pj.freefaculty.org/guides/Rcourse/plot-2/plot-2.pdf`). Graphs that are written directly into a file are more likely to come out correctly, so lets concentrate there.

Run this.

```
x <- rnorm(100)
png(file = "mypng-1.png", height = 800, width = 800)
hist(x, main = "I'm running this and won't see anything on the screen")
dev.off()
```

The dev.off() is a file saver statement. It means "finish the file". In the terminal, you see "null device". Whatever, it worked. A file appears in the current working directory "mypng-1.png". Look at that with an image viewer. Now try another one

```
png(file = "mypng-2.png", height = 300, width = 800, pointsize = 8)
hist(x, main = "I'm running this and won't see anything on the screen")
dev.off()
```

Nothing appears on the screen. R knows the size you want (800 x 800 in pixels). I adjusted the pointsize for fun. You can play with settings.

You don't want a png file? OK, lets try PDF, which is now the recommended format for making graphs that end up in research papers. We used to use postscript, so most of my documentation used to be about EPS. But now, PDF. So run this

```
pdf(file = "mypdf-1.pdf", height = 6, width = 6, onefile = FALSE, paper = "
    special")
hist(x, main = "If there is no PDF at the end of this, I'll be mad")
dev.off()
```

NB: height and width are in inches, NOT pixesl, with the pdf device. I usually only use the png and pdf devices, but I have sometimes tested others, like SVG (scalable vector graphics) and fig (files that can be edited with xfig).

I'm pushing the point that you create the output device, you run the plot commands, and then dev.off(). That works, but it is not useful for planning a plot. You need to see it on the screen, right? R plot functions draw on screen devices by default. As a result, the research work flow is to run the graph commands, look at them on the screen, and then go back and run png(...) or pdf(...) line, then re-run the graphing commands, concluding with dev.off().

I know students ignore this advice, and they think it is a hassle. They create graphs on screen and then use File -> Save As or such from some GUI IDE. Sometimes, they end up with horrible looking graphs. Sizes and shapes turn out wrong. The reason is that the R graph they see on the screen is not seamlessly copied into a file output.

You can reproduce that kind of trouble if you want to. When I'm in a hurry, this is what I do. First, I create an "on screen" device with about the right size. On all platforms, from an R console, this will do so

```
dev.new(height = 7, width = 7)
```

If you happen to be using RStudio, you will get an error message indicating that the function is not available. In that case, on Windows, replace dev.new() with windows(). On Mac, replace it with quartz(). On Linux, X11() is a safe bet. Your system may have other on-screen devices as well. I chose to make a square graph, you don't have to if you don't want to.

After you make the graph look just right, you can save it as a PDF like so:

```
dev.copy(pdf, file = "myfile.pdf", height = 7, width = 7, onefile = FALSE)
dev.off()
```

Back in the olden days of postscript, the onefile = FALSE argument was vital to get the bounding box that put the "encapsulated" into EPS, rather than plain old postscript. I still use it now with PDF, although I don't think it has much effect.

If you want to save the same as a picture image, say a png file with white background, do

```
dev.copy(png, filename = "myfile.png", height = 600, width = 800, bg = "white")
dev.off()
```

In my experience, the bg argument is ignored and the background always turns out transparent.

Note you can also set the pointsize for text. Peter Dalgaard told me that on postscript devices, a point is 1/72nd of an inch. If you expect to shrink an image for final presentation, set a big pointsize, like 20 or so, in order to avoid having really small type.

The dev.copy approach works well OFTEN. Sometimes, however, it is bad. It cuts off the margin labels, it breaks legend boxes. So if you do that, well, you get what you deserve. I've learned this the hard way, but sending out papers with bad graphs and I did not notice until it was too late.

R uses the term "device" to refer to the various graphical output formats. The command dev.off() is vital because it lets the device know you are finished drawing in there. There are many devices representing kinds of file output. You can choose among device types, postscript, png, jpeg, bitmap, etc. (windows users can choose windows metafile). Type

```
?device
```

to see a list of devices R finds on your system. There is a help page for each type of device, so you can run ?png, ?pdf, ?windows to learn more.

After you create a device, check what you have on:

```
dev.list()
```

And, if you have more than one device active, you tell which one to use with dev.set(n), for the n device. Suppose we want the third device:

```
dev.set(3)
```

For me, the problem with this approach is that I don't usually know what I want to print until I see it. If I am using the jpeg device, there is no screen output, no picture to look at. So I have to make the plot, see what I want, turn on an output device and run the plot all over in order to save a file. It seems complicated, anyway.

The pdf and postscript devices have options to output multiple graphs, one after the other. They can either be drawn as new pages inside the single file's output (onefile = TRUE), or one can use a customized file name to ask for it to create one file per graph.

If you use dev.copy(), you may run into the problem that your graph elements have to be resized and they don't fit together the way you expect. The default on-screen device (for me, X11(), for you maybe windows() or quartz()) size is 7in x 7in, but postscript size is 1/2 inch smaller than the usable papersize. That mismatch can spell danger.

The dev.print() function, as far as I can see, is basically the same as dev.copy(), except it has two features. First, the graph is rescaled according to paper dimensions, and the fonts are rescaled accordingly. Due to the problem mentioned above, not everything gets rescaled perfectly, however, so take care. Second, it automatically turns off the device after it has printed/saved its result.

```
dev.print(device = postscript, file = "yourFileName.ps")
```

A default jpeg is 480x480, but you can change that:

```
jpeg(filename="plot.jpg" width = 460, height = 480, pointsize = 12, quality = 85)
```

As of R1.1, the dev.print() and dev.copy2eps() will work when called from a function, for example:

```
ps <- function(file="Rplot.eps", width=7, height=7, ...) {
        dev.copy2eps(file=file, width=width, height=height, ...)
   }
data(cars)
plot(cars)
ps()
```

The mismatch of "size" between devices even comes up when you want to print out an plot. This command will print to a printer:

```
dev.print(height=6, width=6, horizontal=FALSE)
```

You might want to include pointsize=20 or whatever so the text is in proper proportion to the rest of your plot.

One user observed, "Unfortunately this will also make the hash marks too big and put a big gap between the axis labels and the axis title\ldots{}", and in response Brian Ripley observed: "The problem is your use of dev.print here: the ticks change but not the text size. dev.copy does not use the new pointsize: try

```
x11(width = 3, height = 3, pointsize = 8)
x11(width = 6, height = 6, pointsize = 16)
dev.set(2)
plot(1:10)
dev.copy()
```

Re-scaling works as expected for new plots but not re-played plots. Plotting directly on a bigger device is that answer: plots then scale exactly, except for perhaps default line widths and other things where rasterization effects come into play. In short, if you want postscript or pdf, use postscript() or pdf() directly.

The rescaling of a windows() device works differently, and does rescale the fonts. (I don't have anything more to say on that now)

5.3 How to automatically name plot output into separate files (10/04/2001)

Use this notation for the file name: %03d is in the style of old-fashioned fortran format statements. It means use 3 digit numbers like 001, 002, etc.

```
postscript(file="test3-%03d.ps", onefile = FALSE, paper = "special")
```

will create postscriptfiles called test3-001.ps, test3-002.ps and so on (from Thomas Lumley).

```
pdf(file="test3-%03d.pdf", onefile = FALSE, paper = "special")
```

Don't forget

```
dev.off()
```

to finish writing the files to disk. Same works with pdf or other devices.

5.4 Control papersize (15/08/2000)

```
options(papersize = "letter")
```

whereas ps.options is only for the postscript device

5.5 Integrating R graphs into documents: LaTeX and EPS or PDF (20/06/2001)

Advice seems to be to output R graphs in a scalable vector format, pdf or eps. Then include the output files in the LaTeX document.

5.6 "Snapshot" graphs and scroll through them (31/12/2001)

Ordinarily, the graphics device throws away the old graph to make room for the new one.

Have a look to ?recordPlot to see how to keep snapshots. To save snapshots, do

```
myGraph <- recordPlot()
# then
replayPlot(myGraph)
```

to see it again.

On Windows you can choose "history-recording" in the menu and scroll through the graphs using the PageUp/Down keys. (from Uwe Ligges)

5.7 Plot a density function (eg. Normal) (22/11/2000)

```
x <- seq(-3, 3, by = 0.1)
probx <- dnorm(x,0,1)
plot(x, probx, type =``l'')
```

5.8 Plot with error bars (11/08/2000)

Here is how to do it.

```
x <-c(1, 2, 3, 4, 5)
y <-c(1.1, 2.3, 3.0, 3.9, 5.1)
ucl <-c(1.3, 2.4, 3.5, 4.1, 5.3)
lcl <-c(.9, 1.8, 2.7, 3.8, 5.0)
plot(x,y, ylim = range(c(lcl, ucl)))
arrows(x, ucl, x, lcl, length = .05, angle = 90, code = 3)
#or
segments(x,ucl,x,lcl)
(from Bill Simpson)
```

5.9 Histogram with density estimates (14/08/2000)

If you want a density estimate and a histogram on the same scale, I suggest you try something like this:

```
IQR <- diff(summary(data)[c(5, 2)])
dest <- density(data, width = 2*IQR)  # or some smaller width, maybe,
hist(data, xlim = range(dest$x), xlab = "x", ylab = "density",
     probability = TRUE)     # --- this is the vital argument
lines(dest, lty = 2)
(from Bill Venables)
```

5.10 How can I "overlay" several line plots on top of one another? (09/29/2005)

This is a question where terminology is important.

If you mean overlay as in overlay "slides" on a projector, it can be done. This literally superimposes 2 graphs.

Use par("new" = TRUE) to stop the previous output from being erased, as in

```
tmp1 <- plot(acquaint ~ x, type = 'l', ylim = c(0,1), ylab = "average  proportion
    ", xlab = "PERIOD", lty = 1, pch = 1, main = "")
par("new" = TRUE)
tmp2 <- plot(harmony ~ x, type = 'l', ylim = c(0,1), ylab = "average  proportion"
    , xlab = "PERIOD", lty = 2, pch = 1, main = "")
par("new" = TRUE)
tmp3 <- plot(identical ~ x, type = 'l', ylim = c(0,1), ylab = "average proportion
    ", xlab = "PERIOD", lty = 3, pch = 1, main = "")
```

Note the par() is used to overlay "high" level plotting commands.

Here's an example for histograms:

```
hist(rnorm(100, mean = 20, sd = 12), xlim = range(0,100), ylim = range(0, 50))
par(new = TRUE)
hist(rnorm(100, mean = 88, sd = 2), xlim = range(0, 100), ylim = range(0, 50))
```

The label at the bottom is all messed up. You can put the option xlab="" to get rid of them.

There are very few times when you actually want an overlay in that sense. Instead, you want to draw one graph, and then add points, lines, text, an so forth. That is what the R authors actually intend.

First you create the baseline plot, the one that sets the axes values. Then add points, lines, etc. Here are some self-contained examples to give you the idea.

Here's a regression example

```
x <- rnorm(100)
e <- rnorm (100)
y <- 12 + 4*x +13 *e
mylm <- lm(y ~ x )
plot(x, y, main = "My regression")
abline(mylm)
```

From Roland Rau in r-help:

```
hist(rnorm(10000), freq = FALSE)
xvals <- seq(-5,5,length = 100)
lines(x = xvals, y = dnorm(xvals))
```

Here's another example that works great for scatterplots

```
x1 <- 1:10
x2 <- 2:12
y1 <- x1+rnorm(length(x1))
y2 <- .1*x2+rnorm(length(x2))
plot(x1, y1, xlim = range(x1, x2), ylim = range(y1, y2), pch = 1)
points(x2, y2, pch = 2)
(from Bill Simpson)
```

If you just want to plot several lines on a single plot, the easiest thing is to use "matplot" which is intended for that kind of thing. But plot can do it too.

Here's another idea: form a new data.frame and pass it through as y:

```
plot(cbind(ts1.ts, ts2.ts), xlab = "time", ylab = "", plot.type = "single")
```

or better something like:

```
plot(cbind(ts1.ts, ts2.ts), plot.type = "single",
 col=c("yellow", "red"), lty = c("solid","dotted")) #colours and patterns
```

It can also be helpful to contrast 'c(ts1.ts,ts2.ts)' with 'cbind(ts1.ts,ts2.ts)'. (from Guido Masarotto)

For time series data, there is a special function for this:

```
ts.plot(ts1.ts, ts2.ts) # same as above
```

See help on plot.ts and ts.plot (from Brian D. Ripley)

Often, I find myself using the plot command to plot nothing, and then fill in beautiful stuff with text, lines, points, and so forth. If you plot 2 variables with type="n", then the inside of the graph is empty.

5.11 Create "matrix" of graphs (18/06/2001)

If you want a quick "scatterplot" matrix, run pairs() or plot() on a data frame.

Do you want to run individual plot commands and have them placed in an array on the output graph? There are two ways, the basic par(mfrow=?) or par(mfcol=?) or a more advanced layout() function.

Here is how to get started with a 2x3 array of plots:

```
par(mfrow = c(2, 3))
```

And then the next 6 plot/image commands will be laid out in a 2 row x 3 column arrangement.

The layout option can give a powerful set of controls for that kind of thing as well, e.g. (from Paul Murrell):

```
layout(matrix(c(1, 0, 2), ncol = 1), heights = c(1, lcm(5), 1))
plot(x)
box("figure", lty = "dashed")
plot(x)
box("figure", lty = "dashed")
```

I think you have data with like:

```
xx <- data.frame(y = rnorm(100),
                 x1 = as.factor(round(runif(100, 1, 4))),
                 x2 = as.factor(round(runif(100, 1, 4)))
                 )
attach(xx)
by(y, list(x1,x2), plot)
by(y, list(x1,x2), function(x) print(x))
xn <- as.integer(x1)
## either use coplot or par(mfrow =) approach:
coplot(y ~ xn | x2)
##or you use a loop:
par(mfrow = c(2,2))
for( i in unique(as.integer(x1))) plot.default(x2 [as.integer(x1)==i]
                                        , y[as.integer(x1)==i ]
                                        , main = paste("Code:", i)
                                        )
##of course there might be ways to use tapply, lapply etc.
```

(from Peter Malewski. Per Peter Dalgaard's advice, I've replaced usage of codes with as.integer. pj)

And to label the whole page, use the "mtext" function

5.12 Combine lines and bar plot? (07/12/2000)

David James was kind enough to help me out and enlighten me to the fact that the x-scales used by barplot are independent of those from my data.

He also sent me a function which I include below (with a minor modification). This does exactly what I was looking for. Thanks!

```
xbarplot <- function(y, col=par("col"), border = par("fg"), gap=gap, ...){
 ny <- length(y)
```

```
 x <- seq(0,ny)+0.5
 n <- length(x)
 space <- gap * mean(diff(x))
 old <- par(xaxt = "n")
 on.exit(par(old))
 plot(range(x, na.rm=TRUE), range(y, na.rm=TRUE), bty="n",xlab="", ylab = "",
     type = "n", ...)
 rect(x[-n]+space/2, rep(0,length(y)),
 x[-1]-space/2, y, col = col, border = border)
}
(from Dirk Eddelbuettel)
```

5.13 Regression scatterplot: add fitted line to graph (03/20/2014)

The usual recommendation is to use this stanza, which only works when there is only one predictor

```
reg <-lm(y ~ x, data = dat)
plot(y ~ x, data = dat)
# simplest way to add a straight line
abline(reg)
```

If you have more predictors, then you need to think on what you intend to plot.

```
reg <- lm(y ~ x1 + x2 + x3, data = dat)
termplot(reg, se = TRUE, resid = TRUE)
```

Will cycle through the predictors. There's another nice looking approach I suggest in the rockchalk package

```
library(rockchalk)
plotSlopes(reg, plotx = "x2", interval = "confidence")
```

The plotx argument has to be a numeric predictor. That can also draw interaction plots for either numeric or factor moderator variables.

```
library(rockchalk)
plotSlopes(reg, plotx = "x2", modx = "x3", interval = "confidence")
```

I've got a lot of little bells and whistles so users can customize the chosen values of the moderator and other variables. Just run example(plotSlopes) to get the idea.

5.14 Control the plotting character in scatterplots? (11/08/2000)

Plots show dots:

```
plot(rnorm(100),rnorm(100),pch=``.'')
(from Peter Malewski)
```

Plots with numbers indicating values of a third variable:

```
text(tx, ty, label = ch, col = ``red'', bg = ``yellow'', cex = 3)
```

If you specify pch, only the first character is taken as the symbol for your point.

Matt Wiener suggested creating a color vector and then using the value of another value to control both the lable and the color:

```
col.vec <- c("black", "red", "blue")
text(chem.predict[,1:2], labels = km$cluster, color =
col.vec[km$clusters])
```

5.15 Scatterplot: Control Plotting Characters (men vs women, etc)} (11/11/2002)

Remember pch uses integers to select plotting characters.

```
x <- 1:10
y <- 1:10
res <- -4:5
plot(x, y, pch = ifelse(res < 0, 20, 1))
```

If true, use character 20. Else use character 1. Experiment with

different numbers there.

Then note you can create a vector of integers to control symbols, as in

```
g <- 15:24
plot (x,y, pch = g)
(From post by Uwe Ligges 11/7/2002)
```

Here's another cool example from Roger Bivand (11/7/2002),

```
x <- rnorm(25)
y <- rnorm(25)
z <- rnorm(25)
pchs <- c(1, 20)
pchs[(z < 0)+1]
plot(x, y, pch = pchs[(z < 0)+1])
text(x, y, labels = round(z, 2), pos = 1, offset = 0.5)
```

Roger adds, "This "cuts" the z vector at zero, using the convenient slight-of-hand that TRUE and FALSE map to integers 1 and 0, and thus gives the pch argument in plot() or points() a vector of values of indices of the pchs vector. More generally, use cut() to break a numeric vector into class intervals (possibly within ordered())."

5.16 Scatterplot with size/color adjustment (12/11/2002)

```
test<-c(2,6,4,7,5,6,8,3,7,2)
plot(test, type = "n", main = "Color/size test plot", ylab = "Size scale", xlab
   = "Colorscale")
colsiz<-function (yvec) {
  ymin <- min(yvec)
  cexdiv <- max(yvec)/2
  for (i in 1:length(yvec)) {
    nextcex <- (yvec[i] - ymin)/cexdiv + 1
    par(cex = nextcex)
    points(i, yvec[i], type = "p", col = i)
}}
colsiz(test)
```

Note that the size here ranges from 1->2. You can change that by fiddling with the calculation of 'cexdiv'.(from Jim Lemon)

Paul Murrell posted this nice example (11/7/2002):

```
x <- rnorm(50)
y <- rnorm(50)
z <- rnorm(50)
pch <- rep(1, 50)
pch[z < 0] <- 20
cex <- (abs(z)/max(abs(z))) * 4.9 + 0.1
plot(x, y, pch=pch, cex=cex)
```

5.17 Scatterplot: adjust size according to 3rd variable (06/04/2001)

Have a look at:

?symbols

e.g.:

```
symbols(1:10, 1:10, circles = 1:10, inches = 0.1)
(from Uwe Ligges)
```

5.18 Scatterplot: smooth a line connecting points (02/06/2003)

Rado Bonk asked how to smooth a line connecting some points. One answer as, "You may be interested in spline(). For example:

```
x <- 1:5
y <- c(1,3,4, 2.5,2)
plot(x, y)
sp <- spline(x, y, n = 50)
lines(sp)
(from Roger Peng)
```

5.19 Regression Scatterplot: add estimate to plot (18/06/2001)

On my TODO list is a nice formula printed on an angle above a regression line.

For now, all I have are examples of using substitution to take values of variables and insert them into text.

Insert an estimate "that" into a plot title:

```
that <- 1
plot(1:10)
title(substitute(hat(theta)==that, list(that = that)))
(from Brian Ripley)
```

Or, if a value myRsquare is set,

```
text(5, 0.6, as.expression(substitute(R^2==r, list(r=round(myRsquare,3)))))
```

5.20 Axes: controls: ticks, no ticks, numbers, etc (22/11/2000)

Ticks, but no numbers:

```
  plot(1:10, xaxt = "n")
  axis(1, labels = FALSE)
```

This leaves off the x and y titles. You still have the numerical labels

```
plot(x, y, ann = FALSE)
```

If you want a lot of control you can do

```
plot(x, y, ann = FALSE, axes = FALSE)
box()
axis(side = 1, labels = TRUE)
axis(side = 2, labels = TRUE)
mtext("Y", side = 2, line = 2)
mtext("X", side = 1, line = 2)
```

Control range of axes through the graph command itself. This shows the "image" function:

```
x <- 1:50
y <- 1:50
z <- outer(x,y)
image(z)
image(z, xlim = c(0.2,0.5))
image(z, xlim = c(0.2,0.5), ylim = c(0,0.5))
```

The same works with contour(). (from Kaspar Pflugshaupt)

5.21 Axes: rotate labels (06/04/2001)

At one time, apparently par(las=) would do this, as in

```
par(las = 2)
# or
par(las = 3)
```

to make the 90 degrees to axes labels.

More recently, Paul Murrell has observed:

"The drawing of xlab and ylab ignores the par(las) setting. You can override this "sensible" default behaviour if it annoys you. For example, you can stop plot() drawing xlab and ylab by something like plot(ann=FALSE) or plot(xlab="", ylab="") AND you can draw them yourself using mtext(), which does listen to par(las).

A couple of examples of what I mean

```
par(mfrow = c(2,2), mar = c(5.1, 4.1, 4.1, 2.1), las = 0)
plot(0:10, 0:10, type = "n")
text(5, 5, "The default axes")
box("figure", lty = "dashed")
par(las = 1)
plot(0:10, 0:10, type = "n")
text(5, 5, "The user says horizontal text please\n\nbut R knows better!")
box("figure", lty="dashed")
par(las=1, mar=c(5.1, 6.1, 4.1, 2.1))
plot(0:10, 0:10, type="n", ann=FALSE)
mtext("0:10", side=2, line=3)
mtext("0:10", side=1, line=3)
text(5, 5, "The user fights back !")
```

5.22 Axes: Show formatted dates in axes (06/04/2001)

Try package "date", available at CRAN. Example for plotting:

```
library(date)
plot.date(.....)
## if x-values are dates, you can also use plot(...) after loading the data
    package
## because plot is a generic function
(from Uwe Ligges)
```

5.23 Axes: Reverse axis in plot (12/02/2012)

Ross Ihaka says do the following:

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, xlim=rev(range(x)), ylim=rev(range(y)))
```

Another person said try (if you don't want to show the axes):

```
plot(x, -y, axes=FALSE)
axis(1)
axis(2)
```

5.24 Axes: Label axes with dates (11/08/2000)

I have to plot the data set like the following.

```
> x <- c(``05/27/00'',``06/03/00'',``08/22/00'', ...)
> y <- c(10,20,15, ...)
```

You should try the package "date", available at CRAN. Example:

```
library(date)
x <- as.date(c("05/27/2000","06/03/2000","08/22/2000"))
y <- c(10,20,15)
plot(x,y)
(from Uwe Ligges)
```

Here is one using chron.

```
library(chron)
x <- dates(c("05/27/00","06/03/00","08/22/00"))
y <- c(10,20,15)
plot(x, y)
```

which will work if you were in the USA and your first string means 2000

5.25 Axes: Superscript in axis labels (11/08/2000)

Use plotmath, just like in any other text or labels in a plot. Put "y sub i" on left side.

```
plot(c(1:10), ylab = expression(y[i]))
```

5.26 Axes: adjust positioning (31/12/2001)

Question: When I plot them default x axis takes place on the bottom of the chart and the y axis takes place on the left of the chart. Is it possible to make axes pass from the origin.

Maybe this helps (put what you what for x, y):

```
plot(x, y, xaxt = "n", yaxt = "n")
axis(1, pos = 0)
axis(2, pos = 0)
(from Uwe Ligges)
```

5.27 Add "error arrows" to a scatterplot (30/01/2001)

I think using arrows(... , code=3, angle=90, ...) is quite simple, e.g.:

```
x <- rnorm(10)
y <- rnorm(10)
se.x <- runif(10)
se.y <- runif(10)
plot(x, y, pch=22)
arrows(x, y-se.y, x, y+se.y, code=3, angle=90, length=0.1)
arrows(x-se.x, y, x+se.x, y, code=3, angle=90, length=0.1)
(from Emmanuel Paradis)
```

The first arrows() draws the error bars for y, and the second one for x, 'code=3' draws a head at both ends of the arrow, 'angle=' is the angle of the head with the main axis of the arrow, and 'length=' is the length of the head. You can also add usual graphic parameters (col, lwd, \ldots{}).

5.28 Time Series: how to plot several "lines" in one graph? (06/09/2000)

You can run plot and then run the lines function over and over.

```
x <- rnorm(10)
y1 <- rnorm(10)
y2 <- rnorm(10)
y3 <- rnorm(10
plot(x,y1, type="l")
lines(x,y2)
lines(x,y3)
```

matplot is intended to automate this:

```
myys <- cbind(y1, y2, y3)
matplot(x, myys, type ="l")
```

5.29 Time series: plot fitted and actual data (11/08/2000)

```
library(ts)
data(LakeHuron)
md <- arima0(LakeHuron, order=c(2,0,0), xreg=1:98)
plot(LakeHuron)
lines(LakeHuron-md\$resid,col=``red'')
(from Adrian Trapletti)
```

5.30 Insert text into a plot (22/11/2000)

I want this in a plot: "The average temperature is 23.1 degrees."

```
var <- 23.1
text(x,y,paste(``The average temperature is'', var, ``degrees''))
```

5.31 Plotting unbounded variables (07/12/2000)

Graphs blow up if a variable is unbounded, so if you are plotting tan(x), for example:
You probably want to plot something like

```
pmax(--10,pmin(10,tan(x)))
#or
ifelse(abs(tan(x)) > 10, NA, tan(x))
(from Thomas Lumley)
```

5.32 Labels with dynamically generated content/math markup (16/08/2000)

Don Wingate wrote "More generally, what I need to do is dynamically create a vector of math expressions (using the powers of string manipulation), which can then be used as the textual values for a plot function."
Uve Ligges answered: What about something like:

```
numbers <- 1:2  # the dynamical part
my.names <- NULL
for(i in numbers)
   my.names <- c(my.names,
     eval(as.expression(substitute(expression(x[i]^2), list(i=i)))))
dotplot(1:2, labels = my.names)
```

5.33 Use math/sophisticated stuff in title of plot (11/11/2002)

Try help(plotmath).

To integrate values from the program with expressions, use substitute or bquote.

Here are examples in which the plot title is fancied up, but same would work for any other plot text, like xlab, ylab, mtext, or values inside the plot region like text.

Retrieve value of alpha for inclusion in a title:

```
e <- substitute(expression(paste("Power plot of ", x^alpha, " for ",
alpha==ch.a)), list(ch.a=formatC(alpha,wid=1)))
plot(x, x^alpha, main=e)
(from Peter Dalgaard)
```

Want a variable value in the title, as in "Data at the 45o North:

```
lat <- 45
plot(1,main=substitute("Data at "*lat*degree*" North",list(lat=lat)))
(from Thomas Lumley)
```

A Greek letter subscripted in the title

```
plot(1,1,main=expression(gamma[1]=="Threshold 1"))
```

5.34 How to color-code points in scatter to reveal missing values of 3rd variable? (15/08/2000)

```
plot(x,y,color=ifelse(is.na(z),``red'',``black''))
(from Peter Dalgaard)
```

```
plot(x[!is.na(z)],y[!is.na(z)],xlim=range(x),ylim=range(y))
points(x[is.na(z)],y[is.na(z)],col=2)
(from Ross Darnell)
```

5.35 lattice: misc examples (12/11/2002)

The lattice library came along recently, I've not explored it much. But here is a usage example, emphasizing the aspect setting from Paul Murrell:

```
library(lattice)# 10 rows
x <- matrix(rnorm(100), ncol=10)
lp1 <- levelplot(x, colorkey=list(space="bottom"),aspect = 10/10)
print(lp1)
# 5 rows -- each row taller than in previous
plotx <- matrix(rnorm(50), ncol=5)
lp2 <- levelplot(x, colorkey=list(space="bottom"),aspect=5/10)
print(lp2)
```

This one from Deepayan Sarkar (deepayan@stat.wisc.edu) is especially neat.

```
library(lattice)
xyplot(rnorm(100) ~ rnorm(100) | gl(2, 50),       strip = function(factor.levels,
    ...) { strip.default(factor.levels = expression(1 * degree, 2 * degree), ...)
    })
```

5.36 Make 3d scatterplots (11/08/2000)

Type

?persp

?contour

to see about features in the base package.

There is a scatterplot3d package on CRAN.

See also:

http://www.statistik.uni-dortmund.de/leute/ligges.htm (from Uwe Ligges)

5.37 3d contour with line style to reflect value (06/04/2001)

Suppose you want a contour plot solid contour lines for positive values and dotted contour lines for negative values. The trick is to specify the levels and to use the add= argument.

```
x <- seq(-1, 1, length=21)
f <- function(x,y) (x^2 + y^2) / 2 - 0.5
z <- outer(x, x, f)
contour(z, levels = seq(-0.5, -0.1, by = 0.1), lty = "dotted")
contour(z, levels = 0, lty = "dashed", add = TRUE)
contour(z, levels = seq(0.1, 1, by = 0.1), add = TRUE)
(from Ross Ihaka)
```

5.38 Animate a Graph! (13/08/2000)

It is quite easy to do with ImageMagick (www.imagemagick.org), which can be installed on most OSes. I tried this simple sequence and it worked beautifully.

In R, create 15 histograms:

```
for(i in 1:5) {
 jpeg(paste("fig", i, ".jpg", sep = ""))
 hist(rnorm(100))
 dev.off()
}
```

Then from the command line (I tried it using Linux, though it should be the same on any platform):

```
% convert -delay 50 -loop 50 fig*.jpg animated.gif
```

This created animated.gif, a nice animation of my sequence of files. You can control the timing of the animation by playing with -delay and -loop. (from Matthew R. Nelson, Ph.D.)

5.39 Color user-portion of graph background differently from margin (06/09/2000)

Use "par" to find out where the boundaries/axes are, and "rect" to draw the picture,

The 6th plot in demo(graphics) does the same thing.

```
## code for coloring background
x <- runif(10)
y <- runif(10)

plot(x,y,type="n")
u <- par("usr")
rect(u[1],u[3],u[2],u[4],col="magenta")
points(x,y)
(from Ben Bolker)
```

(pj: see next because it includes a usage of do.call)

I think one is stuck with things like

```
plot(2:10, type="n")
do.call("rect", c(as.list(par("usr")[c(1,3,2,4)]),list(col="pink")))
points(2:10)

## or for the middle line, somewhat simpler
bb <- par("usr")
rect(bb[1], bb[3], bb[2], bb[4], col="pink")
(from Peter Dalgaard )
```

5.40 Examples of graphing code that seem to work (misc) (11/16/2005)}

Tito de Morais Luis emailed this example to me. It shows several useful and widely-used "tricks" for customizing plots

```
# truncated y-axis plot
# from Wolfgang Viechtbauer R-help archive available at
# http://tolstoy.newcastle.edu.au/~rking/R/help/03a/7415.html
y <- c(140, 120, 110, 108, 104)
plot(x, y, yaxt="n")
axis(side = 2, at = c(100, 105, 110, 115, 120, 140), labels = c(100, 105, 110,
    115, 120, 1000))
rect(0, 130, 1, 131, col = "white", border = "white")
par(xpd=TRUE)
lines(x = c(0.7,1), y = c(130, 130))
lines(x = c(0.7,1), y = c(131, 131))
# My use is based on the above.
# It allows the plot on the same graph of water level curves from 2 African
    man-made lakes
level <- data.frame(Selengue=c(343.9, 347.5, 348.5, 348.7, 348.4, 347.8, 347.2,
    346.3, 345.2, 343.9, 342.3, 341.5), Manatali=c(203.0, 207.0, 208.0, 208.5, 208
    .0, 207.7, 207.0, 206.0, 204.5, 203.0, 202.0, 201.0))
rownames(level) <- c("Aug", "Sep", "Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr
    ", "May", "Jun", "Jul")
lev <- data.frame(Selengue=level[,1]-130,Manantali=level[,2])
plot(lev[,1],ylim=c(200,220),xlab="",ylab="", yaxt="n", xaxt="n", type='b', pch
    =19)
par(new=TRUE)
plot(lev[,2],ylim=c(200,220),xlab="Month",ylab="Water level (m)", yaxt="n", xaxt=
    "n", type='b', pch=21)
axis(side=2, at=c(200, 205, 210, 215, 220), labels=c(200, 205, 210, 245, 350))
axis(side=1, at=c(1,2,3,4,5,6,7,8,9,10,11,12), labels=rownames(lev))
rect(-0.5, 210.8, 1, 211.8, col="white", border="white")
par(xpd=TRUE)
lines(x=c(0.4,0.7), y=c(210.8, 210.8))
lines(x=c(0.4,0.7), y=c(211.8, 211.8))
```

```
x <- c(1,2,3,4); y1 <- c(1,2,3,4); y2 <- c(2,2,2,2)
Fred <- c(1,2)

postscript(file="try2.ps")
 plot(x,y1, type="l")
 lines(x,y2,lty="33")
 legend(1,4, c("y1","y2"), lty=Fred)
dev.off()
```

Plot sin and cos

```
layout.my <- function (m, n) {
    par(mfrow = c(m, n))
}

x <- 0:12566 / 1000    # Range from 0 to 4*pi
```

```
layout.my( 1, 2 )
plot( sin(x), type = "l",
      xaxs = "i", yaxs = "i", axes = F,
xlab = "x", ylab = "sin(x)",
      main = "Y = sin(x), x = [ 0, 720 ]"
)
axis( 2, at = seq( -1, 1, by=1 ),las = 2 )
box(lty="dotted")
abline( h = 0, lwd = 1 )

plot( cos(x), type = "l",
      xaxs = "i", yaxs = "i", axes = F,
      xlab = "x", ylab = "cos(x)",
      main = "Y = cos(x), x = [ 0, 720 ]"
    )
axis( 2, at = seq( -1, 1, by=1 ),las = 2 )
box(lty="dotted")
abline( h = 0, lwd = 1 )
(from Ko-Kang Wang)
```

Plot a regular polygon. Below is a function that generates regular polygons, filled or borders, of n sides (n>8 => circle), with "diameter" prescribed in mm, for use alone or with apply.

```
ngon <- function (xydc, n=4, type=1)
  # draw or fill regular polygon
  # xydc a four element vector with
#   x and y of center, d diameter in mm, and c color
  # n number of sides of polygon, n>8 => circle
  #   if n odd, vertex at (0,y), else midpoint of side
  # type=1 => interior filled, type=2 => edge
# type=3 => both
{
  u <- par("usr")
p <- par("pin")
  d <- as.numeric(xydc[3])
  inch <- d/25.4
  rad <- inch*((u[2]-u[1])/p[1])/2
  ys <- inch*((u[4]-u[3])/p[2])/2/rad
  if (n > 8) n <- d*4 + 1
  th <- pi*2/n
  costh <- cos (th)
  sinth <- sin (th)
  x <- y <- rep (0,n+1)
  if (n %% 2) {
    x[1] <- 0
    y[1] <- rad
    }
  else {
x[1] <- -rad*sin(th/2)
    y[1] <-  rad*cos(th/2)
}
  for (i in 2:(n+1)) {
    xl <- x[i-1]
    yl <- y[i-1]
    x[i] <- xl*costh - yl*sinth
    y[i] <- xl*sinth + yl*costh
    }
  x <- x + as.numeric(xydc[1])
  y <- y*ys + as.numeric(xydc[2])
  if (type %% 2) polygon (x,y,col=xydc[4],border=0)
  if (type %/% 2) lines (x,y,col=xydc[4])
  invisible()
}  (from Denis White)
```

```
plot(1:1000, rnorm(1000), axes=FALSE)
# plot with no axes
axis(1, at = seq(0, 1000, by = 100)) # custom x axis
axis(2) # default y axis
box() # box around the plot
(from Ross Ihaka)
```

Here is an example of drawing a cubic with axes in the middle of the plot.

```
#  Draw the basic curve (limits were established by trial and error).
#  The axes are turned off blank axis labels used.

curve(x * (2 * x - 3) * (2 *  x + 3), from = -1.85, to = 1.85,
              xlim = c(-2, 2), ylim = c(-10, 10),
              axes = FALSE, xlab = "", ylab = "")

#  Draw the axes at the specified positions (crossing at (0, 0)).
#  The ticks positions are specified (those at (0,0) are omitted).
#  Note the use of las=1 to produce horizontal tick labels on the
#  vertical axis.

axis(1, pos = 0, at = c(-2, -1, 1, 2))
axis(2, pos = 0, at = c(-10, -5, 5, 10), las = 1)

#  Use the mathematical annotation facility to label the plot.

title(main = expression(italic(y == x * (2 * x - 3) * (2 * x + 3))))
(from Ross Ihaka)
```

Question: is it possible to shade the 3d surface like a contour plot? i.e. black for large z, white for small z, say

Answer:

```
# Create a simple surface f(x,y) = x^2 - y^2
nx <- 21
ny <- 21
x <- seq(-1, 1, length = nx)
y <- seq(-1, 1, length = ny)
z <- outer(x, y, function(x,y) x^2 - y^2)

#  Average the values at the corner of each facet
#  and scale to a value in [0, 1].  We will use this
#  to select a gray for colouring the facet.

hgt <- 0.25 * (z[-nx,-ny] + z[-1,-ny] + z[-nx,-1] + z[-1,-1])
hgt <- (hgt - min(hgt))/ (max(hgt) - min(hgt))

#  Plot the surface with the specified facet colours.

persp(x, y, z, col = gray(1 - hgt), theta = 35)
persp(x, y, z, col = cm.colors(10)[floor(9 * hgt + 1)], theta = 35)
(from Ross Ihaka)
```

Polygon!

```
y <- c(1.84147098480790, 1.90929742682568, 1.14112000805987,
 0.243197504692072, -0.958924274663133,0.720584501801074,
 1.65698659871879, 1.98935824662338, 1.41211848524176,0.45597888911063)

 plot(y,ylim=c(0.0000000001,2),type="l",log="y")

 x <- 1:10
 temp <- which(y<0)
 polygon(x[-temp], y[-temp], col="red")
(from Uwe Ligges)
```

A use of matplot side-by-side output

```
p <- 1:3
u <- matrix(c(1,1,1,2,2,2,3,3,3),3,)
r <- p*u
X11(height=3.5,width=7)
par(mfcol=c(1,3),cex=1.5,mex=0.6, mar=c(5,4,4,1)+.1)
matplot(p,r,type="b",main="A",col="black")
matplot(log(p),log(r),type="b",main="B",col="black")
r <- p+u
matplot(p,r,type="b",main="C",col="black")
(from Peter Dalgaard, June 11, 2001)
```

6 Common Statistical Chores

6.1 Crosstabulation Tables (01/05/2005)

A crosstab facility xtabs() was introduced in R1.2. Read all about it in help. table is there too. These give counts, not percentages. There are special functions for that. Here's code that creates a table and then calls "prop.table" to get percents on the columns:

```
> x<- c(1,3,1,3,1,3,1,3,4,4)
> y <- c(2,4,1,4,2,4,1,4,2,4)
> hmm <- table(x,y) > prop.table(hmm,2) * 100
```

If you want to sum the column, there is a function "margin.table()" for that, but it is just the same as doing a sum manually, as in:

apply(hmm, 2, sum)

which gives the counts of the columns.

If you have an old R, here is a "do it yourself" crosstab

```
>count <- scan()
65 130 67 54 76 48 100 111 62 34 141 130 47 116 105 100 191 104
>s <- c("Lo", "Med", "Hi")
>satisfaction <- factor( rep(c(s,s), rep(3,6)), levels =>c("Lo","Med","Hi"))
>contact <- factor( rep(c("Low","High"), rep(9,2)),levels =>c("Low","High"))
>r <- c("Tower", "Flat", "House")
>residence <-factor(rep(r,6))
>tapply(count,list(satisfaction,contact,residence),sum)
```

On 2005–01–05, Brian Ripley proposed one way to make a table that has the "value" being counted in the first column (for tables of counts that works nicely for non-factor variables)

```
>xx <- unique(x)
>rbind(vals=xx, cnts=tabulate(match(x, xx)))
```

Check this out! User wants to count outcomes by categories and make a single table list. Thomas Lumley says (2005–01–04), "table() will count frequencies, and interaction() will create a variable with all combinations of a set of variables, so something like

```
>table(interaction(f1,f2))
```

works."

6.2 t-test (18/07/2001)

Index the vector by group:

```
t.test(x[sex==1], x[sex==2])
```

should do the trick. You could try:

```
summary(aov(sex~group))
```

which is equivalent but assumes equal variances. (from Mark Myatt)

The power of the test can be calculated:

power.t.test()

see help(power.t.test)

6.3 Test for Normality (31/12/2001)

Use shapiro.test(x) for testing whether x comes from a normal distribution. X must be a vector or variable.

6.4 Estimate parameters of distributions (12/02/2012)

The most commonly recommended function to estimate parameters is fitdistr() in the MASS package.

See also gnlr() and gnlr3() in Jim Lindsey's gnlm package at `http://www.luc.ac.be/~jlindsey/rcode.html`.

A worked-out example is given as xmp06.12 in the Devore5 package. Devore5 is a collection of examples for Devore's "Probability and Statistics for Engineering (5th ed)" that was put together by Doug Bates

```
library(Devore5)
?Devore5::xmp06.12
example(xmp06.12)    # run the example to see the result
```

With R 2.14, I'm having trouble with the namespace for Devore5, so we may need to contact the author.

6.5 Bootstrapping routines (14/08/2000)

The boot package is distributed with R.

6.6 BY subgroup analysis of data (summary or model for subgroups)(06/04/2001)

Question: I want summary information by subgroup.

Answer: This can be done with tapply, or the by() function, which is a "wrapper" (that means "simplifying enhancement") of tapply.

Use like so. If you have groups defined by a factor A, to get summaries of variables x through z in dataframe "fred",

```
>attach(fred)
>by(fred[,x:z], A, summary)
```

You can make a list of factors for a multidimensional set of groups, and many different functions can be put in place of "summary". See the by documentation.

Before by, we did it like this, and you may need it for some reason.

Get the average of a variable "rt" for each subgroup created by combinations of values for 3 factors.

```
tapply(rt,list(zf, xf, yf), mean, na.rm=TRUE)
```

alternatively,

```
ave(rt, zf, xf, yf, FUN=function(x)mean(x,na.rm=TRUE))
(from Peter Dalgaard)
```

7 Model Fitting (Regression-type things)

7.1 Tips for specifying regression models (12/02/2002)

1. Rather than

```
lm(dataframe$varY ~ dataframe$varX)
```

do

```
lm(varY ~ varX, data = dataframe)
```

2. To fit a model with a "lagged" variable, note there is no built in lag function for ordinary vectors. lag in the ts package does a different thing. So consider:

```
mylag <- function(x,d=1) {
n <- length(x)
c(rep(NA,d),x)[1:n]
}
lm(y ~ mylag(x))
```

7.2 Summary Methods, grabbing results inside an "output object"

(**17/02/2002**)
Brian Ripley: By convention summary methods create a new object of class print.summary.foo which is then auto-printed.
Basically, you find out what's in a result, then grab it. Suppose a fitted model is "fm3DNase1":

```
> names(summary(fm3DNase1))[1] "formula"      "residuals"    "sigma"        "df"
             "cov.unscaled"[6] "correlation"  "parameters"  > summary(fm3DNase1)$
   sigma[1] 0.01919449> summary(fm3DNase1)$parametersEstimate Std. Error  t value
       Pr(>|t|)Asym 2.345179 0.07815378 30.00724 2.164935e-13xmid 1.483089 0
   .08135307 18.23027 1.218523e-10scal 1.041454 0.03227078 32.27237 8.504308e-14#
    for the standard errors:> summary(fm3DNase1)$parameters[,2]      Asym
   xmid       scal 0.07815378 0.08135307 0.03227078
```

(from Rashid Nassar)
The t-values are
summary(fm3DNase1)$coefficients[, "t value"]
and the standard errors are
summary(fm3DNase1)$coefficients[, "Std. Error"]
You can also use
vcov(fm3DNase1)
to get the estimated covaraince matrix of the estimates.

7.3 Calculate separate coefficients for each level of a factor (22/11/2000)

grp is a factor. Then:

```
R <- lm(y ~ grp:x, data=foo)
Call: lm(formula = y ~ grp:x, data = foo)
Coefficients: (Intercept) grpa.x grpb.x grpc.x
--0.80977 0.01847 0.13124 0.14259
( Martyn )
```

7.4 Compare fits of regression models (F test subset B's =0) (14/08/2000)

```
summary(lm.sml <- lm(y ~ x1))
  summary(lm.big <- lm(y ~ x1 + x2 + x3 + x4))
  anova(lm.sml, lm.big)
```

now will do a test of b2 = b3 = b4 = 0

Look at

```
demo(lm.glm)
```

where models "l1" and "l0" are compared that way. (from Martin Maechler)

Here's another one along the same lines. Bill Venables (Feb.2,00) said the philosophy is "You fit a larger model that contains the given model as a special case and test one within the other." "Suppose you wonder if a variable x Suppose you have only one predictor with repeated values, say x, and you are testing a simple linear regression model. Then you can do the test using

```
inner.mod <- lm(y ~ x, dat)
outer.mod <- lm(y ~ factor(x), dat)
anova(inner.mod, outer.mod)
```

Test for lack of fit done. If you have several predictors defining the repeated combinations all you need do is paste them together, for example, and make a factor from that. (from Bill Venables)

Suppose your data frame x has some column, say, ID, which identifies the various cases, and you fitted

```
fit1 <- lm(y ~ rhs, data=df)
```

Now do

```
fit2 <- lm(y ~ factor(ID), data=df)
anova(fit1, fit2, test="F")
```

e.g.

```
set.seed(123)
df <- data.frame(x = rnorm(10), ID=1:10)[rep(1:10, 1+rpois(10, 3)), ]
df$y <- 3*df$x+rnorm(nrow(df))
fit1 <- lm(y ~ x, data=df)
fit2 <- lm(y ~ factor(ID), data=df)
anova(fit1, fit2, test="F")
```

there is an R package lmtest on CRAN, which is full of tests for linear models. (from Brian Ripley)

The models with factor() as the indep variable are the most general because they individually fit each category, whereas using the variable X assumes linearity (from me!)

Here's another example. "I want to know how much variance of pre/post-changes in relationship satisfaction can be explained by changes in iv1 and iv2 (dv ~ iv1 + iv2), by changes in iv3 and iv4 (dv ~ iv3 + iv4) and I want to know wether a model including all of the iv's explains a significant amount of variance over these two models (dv ~ iv1 + iv2 + iv3 + iv4)."

```
model.0  <- lm(dv ~ 1)
model.A  <- lm(dv ~ iv1 + iv2)
model.B  <- lm(dv ~ iv3 + iv4)
model.AB <- lm(dv ~ iv1 + iv2 + iv3 + iv4)

anova(model.AB, model.A, model.0)
anova(model.AB, model.B, model.0)
```

(from Ragnar Beer)

7.5 Get Predicted Values from a model with predict() (11/13/2005)

Suppose you fit a model "lm1". That model is supposed to include methods fitted() and predict().

If you want the predicted value for each nonmissing case, do

```
> predict(lm1)
```

predict has a lot more power because it can tell you the predicted values for a certain set of inputs into the model. Give it a dataframe as the second argument. That new data frame must have (at least) all of the variables in the original model. For example,

```
x<-c(1,3,2,1,4,1,4,NA)
y<-c(4,3,4,1,4,1,4,2)
mod1 <-m(y~x)
testdataframe<-data.frame(x=c(1)) predict(mod1,newdata=testdataframe)
```

Or:

```
> lot <- c(30,20,60,80,40,50,60,30,70,60)
> hours <-c(73,50,128,170,87,108,135,69,148,132)
> z1 <- lm(hours~lot)
> newdf <- data.frame(lot=80)
> predict(z1,newdata=newdf,interval="confidence",level=0.90)
      fit       lwr       upr[1,] 170 166.9245 173.0755
```

Please note, there is a difference between "confidence" and "prediction" in this command. One is the confidence interval of the point estimate, the other is the confidence interval for a prediction about an individual case.

Read carefully the documentation on your model. If you use glm to fit a model, you can read the docs for it with

```
?predict.glm
```

. In particular, the predict method for glm can calculate various types of predictions, the value of the "linear predictor" X(b-hat) or it can give the transformed value, such as the probability in a logistic regression. The option there is type="link" for the X(b-hat) or type="response" for the probability that corresponds to X(b-hat). Know what I mean, either X(b-hat) or 1/(1-exp(-X(b-hat))).

Lately, I've been teaching regression and it is really really hard for newcomers to get predicted values for "interesting" cases. We've gone through all kinds of tedious routines to calculate the "new data frame" so that the output from predict is interesting and useful for graphs.

Suppose your old data frame is "olddf" and you want to create a new data frame for a problem with predictors X1,X2,X3,X4. One idea is to set X2, X3, X4 at their means and then allow X1 to vary from 1 to 10. This works:

```
mymod1 <- glm(Y~X1+X2+X3+X4, data=olddf, family=binomial(link=logit))
newdf <- data.frame(X1=1:10,X2=mean(olddf$X2),X3=mean(olddf$X3),X4=mean(olddf$X4)
    )
p1 <- predict(mymodl1,newdata=data.frame,type="response")
plot(1:10,p1, main="Logistic demonstration graph with X2,X3,X4 at mean values")
```

That provides a plot that people like.

It is impossibly tedious to keep re-generating data frames to explore the variables. There are some shortcuts. In R's base, there is a function called expand.grid (more on that in a moment). In the Design library from Prof. Harrell, the "datadist" procedure is used to create information about the variables that is used in automatically creating that kind of plot. However, sometimes, the plots are not exactly what one might like. But usually they are pretty good, at least as a

starting point. To make a plot demonstrating X1, it will set other variables to their means if they are numeric and to their modes if they are not numeric.

In the end, however, a person may want to calculate predicted values for a variety of different outputs. Here is where the function "expand.grid" comes in very handy. If one specifies a list of values to be considered for each variable, then expand.grid will create a "mix and match" data frame to represent all possibilities.

```
> x <- c( 1, 2, 3, 4 )
> y <- c( 22.1, 33.2, 44.4 )
> expand.grid(x,y)
   Var1 Var2
1     1 22.1
2     2 22.1
3     3 22.1
4     4 22.1
5     1 33.2
6     2 33.2
7     3 33.2
8     4 33.2
9     1 44.4
10    2 44.4
11    3 44.4
12    4 44.4
```

So, if you create the new data frame with

```
newdf <- expand.grid(X1,X2,X3,X4)
```

, then put that as newdata into predict and one easily obtains all of the desired information.

Note many models have a "fitted" attribute and there is the accessor function fitted. If you use fitted() on a model, it may not return what you expect. For lm() it gives the same as predict, but not for glm(). The default for predict() gives the "linear predictor", while fitted gives the value of "mu" (inverse link applied to linear predictor). fitted(mod) equals predict(mod,ndf,type="response").

```
x <- rnorm(100)
y <- rnorm(100)
mylm<-glm((y>0)~x,family=binomial)
> attributes(mylm)
$names
 [1] "coefficients"      "residuals"         "fitted.values"
 [4] "effects"           "R"                 "rank"
 [7] "qr"                "family"            "linear.predictors"
[10] "deviance"          "aic"               "null.deviance"
[13] "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"
[19] "converged"         "boundary"          "model"
[22] "call"              "formula"           "terms"
[25] "data"              "offset"            "control"
[28] "method"            "contrasts"         "xlevels"
> predict(mylm)
        1         2         3         4         5         6         7         8
        9        10
0.7341672 0.3246603 0.2955893 0.4058782 0.5045060 0.5282107 0.3429243 0.1159356 0
    .3394576 0.4889709
> fitted(mylm)
    1         2         3         4         5         6         7         8    9
           10
0.6757191 0.5804596 0.5733639 0.6000991 0.6235177 0.6290657 0.5849007 0.5289515 0
    .5840588 0.6198640
> predict(mylm,type="response")
        1         2         3         4         5         6         7         8
                  9         10
```

```
0.6757191 0.5804596 0.5733639 0.6000991 0.6235177 0.6290657 0.5849007 0.5289515 0
    .5840588 0.6198640
```

To make a table showing how factors affect the fitted or predicted values, Brian Ripley says (2003–11–29) ":
allergy$fitted <- fitted(allergy.fit.main.2int) xtabs(fitted ~ t + f + c, data=allergy)
and you permute the dimensions by reordering the factors.... xtabs generates a call to table() which could be used directly.
In this case there were no repeated (and no missing) combinations of factors: if there has been just predict the model at all combinations and apply xtabs/table to the predictions."

7.6 Polynomial regression (15/08/2000)

Use the I() function to protect the 2 and 3 from being evaluated as part of the formula, ie.

```
formula = Response ~ Var1 + I(Var1^2) + I(Var1^3)
```

(from Douglas Bates)
I was doing this in a practical class yesterday. There is another way,

```
Response ~ poly(Var1, 3)
```

which fits the same cubic model by using orthogonal polynomials. The fitted values will be the same, but the coefficients are those of the orthogonal polynomials, not the terms in the polynomial. You might like to compare the two approaches. (from Brian D. Ripley)

7.7 Calculate "p" value for an F stat from regression (13/08/2000)

a$fstatistic returns the actual f-test and df, but not the p-value. Get a linear model, "grab" the f part of the output, use the f distribution (pf).

```
foo <- lm(y~x)
 a <- summary(foo)
 f.stat <- a$fstatistic
 p.value <- 1-pf(f.stat["value"],f.stat["numdf"],f.stat["dendf"])
```

(from Guido Masarotto)

7.8 Compare fits (F test) in stepwise regression/anova (11/08/2000)

Begin with:

```
data(stackloss)
 fm <- lm(stack.loss ~ ., data=stackloss)
 anova(fm)
```

Add more models:

```
fm2 <- update(fm1, . ~ . - Water.Temp)
fm3 <- update(fm2, . ~ . - Air.Flow)
anova(fm2, fm1)
anova(fm3, fm2)
```

Note that the SSqs are all the same, but the sequential table compares them to the residual MSq, not to the next-larger model (from Brian D. Ripley)

7.9 Test significance of slope and intercept shifts (Chow test?)

(**22/11/2000**)

Mark M. Span asked this: I have a four-parameter, biexponential model. What I want to evaluate is whether the parameter values differ between (groups of) tasks.

```
myfunc    <-formula(x ~ a*exp(-b*age) + (c*exp(d*age)) )
```

both x (representing response time) and age are variables in the current scope. At the moment I fit the model for each task seperately. But now I cannot test the parameters for equality.

Answer: Use a model parametrizing a,b,c by groups and one without, and use anova to compare the models. A worked example is in V&R3, page 249. (from Brian Ripley)

7.10 Want to estimate a nonlinear model? (11/08/2000)

Use the nls library

```
library(nls)
        ... get data
        model.fit <- nls(y ~ exp(x))
        plot(model.fit)
```

7.11 Quasi family and passing arguments to it. (12/11/2002)

(from Halvorsen) I have the following simple function:

```
testcarfunction(pow){
ob <-glm(Pound~CG+Age+Vage, data=car, weights=No, subset=No>0, family = quasi(
    link=power(pow),var=mu^2))
 deviance(ob)
}
```

But trying to run it gives:> Error in power(pow) : Object "pow" not found

A more reliable version (replacement) is

```
eval(substitute(glm(Pound~CG+Age+Vage,data=car,    weights=No subset=No>0,   family
   =quasi(link=power(pow),var=mu^2)),    list(pow=pow)))
(from Thomas Lumley)
```

7.12 Estimate a covariance matrix (22/11/2000)

var(X) or vcov(X), depending on what you are asking for. If you just want the standard errors of a fitted model, don't forget you can take the results from summary() as described above.

The vcov() function is a generic for which many types of fitted models have implemented methods. lm and glm work just fine, and many other types will as well. So try

```
m1 <- lm(y ~ x1 + x2 + x3, data=dat)
vcov(m1)
## Note the standard errors in summary(m1) match this:
sqrt(diag(vcov(m1)))
```

too. As it's generic, it works for most fitted model objects.

One way to grab standard errors. Or, to just save the diagnoal "standard errors" of coefficients, do

```
stdErrVector <- sqrt(diag(vcov(lm.D9)))
```

Or, the hard way,

```
> example(lm)
> summary(lm.D9)
> lm.D9.sum <- summary(lm.D9)
> coef(lm.D9.sum)
             Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 5.032  0.2202177 22.85012 9.547128e-15
groupTrt    -0.371  0.3114349 -1.19126 2.490232e-01
> coef(lm.D9.sum)[ ,"Std. Error"]
(Intercept)    groupTrt
0.2202177   0.3114349
```

7.13 Control number of significant digits in output (22/11/2000)

The global options can be set and many functions will obey them. Check ?options or try

```
options(digits=3)
```

Many functions have their own digits option, such as summary

```
> summary(c(123456,1,2,3), digits=7)
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.00 1.75 2.50 30865.50 30866.25 123456.00
```

and you get what you want. In most cases the number of "significant digits" printed out by summary methods in R is truncated to 3 or 4. (from Robert Gentleman)

If that doesn't produce the desired result, use the round function to force the result to the desired size.

```
> x <- rnorm(7)
> round(x, 2)
[1] -0.01 -0.86  0.20  1.25  0.36  0.77 -0.52
> mean(x)
[1] 0.171956
> mx <- mean(x)
> round(mx, 2)
[1] 0.17
```

7.14 Multiple analysis of variance (06/09/2000)

```
aov(cbind(y1,y2,y3)~x)
```

7.15 Test for homogeneity of variance (heteroskedasticity) (12/02/2012)

Bartlett's test, I like (personally). SPSS now uses Levene's test, some say incorrectly and non-robustly. lmtest package offers many heteroskedasticity estimators. There is an implementation of Levene's test in the car package.

The test "does an anova on a modified response variable that is the absolute value of the difference between an observation and the median of its group (more robust than Levene's original choice, the mean)". (from Brian Ripley)

Incidentally, if you insist on having Levene's calculation, Brian Ripley showed how it can be done manually:

```
>Levene <- function(y, group){
     group <- as.factor(group)  # precautionary
     meds <- tapply(y, group, median)
     resp <- abs(y - meds[group])
     anova(lm(resp ~ group))[1, 4:5]
 }
```

```
> data(warpbreaks)
> attach(warpbreaks)
> Levene(breaks, tension)
      F value   Pr(>F)
  group   2.818 0.06905
```

"I could (and probably would) dress it up with a formula interface, but that would obscure the simplicity of the calculation."

7.16 Use nls to estimate a nonlinear model (14/08/2000)

```
x <- runif(100)
w <- runif(100)
y <- x^2.4*w^3.2+rnorm(100,0,0.01)
plot(x,y)
plot(w,y)
library(nls)
fit <- nls(y~x^a*w^b,start=list(a=2,b=3))
l.est <- lm(log(y) ~ log(x)+log(w)-1)
fit <- nls(y~x^a*w^b,start=list(a=coef(l.est)[1],b=coef(l.est)[2]))
```

Error is a fairly common result of starting with parameter values that are far away from the true values, or have very different scales, so that the numeric-derivative part of nls fails. Various solutions are to do an approximate least-squares regression to start things off (also the "self-starting" models in the nls package), or to provide an analytic derivative. (all from Ben Bolker)

7.17 Using nls and graphing things with it (22/11/2000)

If you use the nls function in R to fit the model in the parameterization that Guido described, you can graphically examine the profile likelihood with

```
pr <- profile(nlsObject)
 plot(pr)
```

For an example, try

```
library(nls)
 example(plot.profile.nls)
```

(from Douglas Bates)

7.18 $-2Log(L)$ and hypo tests (22/11/2000)

This is about the polr package, but the general point is that to compare likelihood models, you have to use your head and specify the comparison. Brian Ripley said:

As in the example on the help page

```
options(contrasts=c("contr.treatment", "contr.poly"))
     data(housing)
     house.plr <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
     house.plr0 <- polr(Sat ~ 1, weights = Freq, data = housing)
```

Note that –2 LOG L is not actually well-defined as it depends on the grouping of the data assumed, and so is not a statistic. I assume that you want differences of that quantity, as in

```
house.plr0$deviance - house.plr$deviance
169.7283
```

7.19 logistic regression with repeated measurements (02/06/2003)

Hiroto Miyoshi described an experiment with 2 groups and pre/post measurement of a dichotomous dependent variable. Frank Harrell had this excellent response:

"There are several ways to go. GEE is one, random effects models another. One other approach is to install the Hmisc and Design packages (http://hesweb1.med.virginia.edu/biostat/s) and do (assume id is the unique subject identifier):

```
f <- lrm(y ~ x1 + x2*x3 + ..., x=TRUE, y=TRUE) # working independence model
g <- robcov(f, id)          # cluster sandwich variance adjustment
h <- bootcov(f, id, B=100) # cluster bootstrap adjustment
summary(g)  # etc.
```

7.20 Logit (06/04/2001)

Compute the Odds Ratio and its confidence intervall, from a logistic model (glm(.....,family=binomial....). I show a simple function to do this in my introductory notes available from: http://www.myatt.demon.co.uk Basically it is:

```
lreg.or <- function(model){
  lreg.coeffs <- coef(summary(salex.lreg))
  lci <- exp(lreg.coeffs[ ,1] - 1.96 * lreg.coeffs[ ,2])
  or <- exp(lreg.coeffs[ ,1])
  uci <- exp(lreg.coeffs[ ,1] + 1.96 * lreg.coeffs[ ,2])
  lreg.or <- cbind(lci, or, uci)
  lreg.or
}
```

(from Mark Myatt)

7.21 Random parameter (Mixed Model) tips (01/05/2005)

Since the lme/nlme support in R is cutting-edge and Doug Bates is an author and frequent R contributor, this is one of the packages that is worth looking at. When I see useful tidbits, I'm putting them here:

Test whether one should use a model with mixed effects or just a plain old linear model:

Bates wrote (2005–01–05) "I would recommend the likelihood ratio test against a linear model fit by lm. The p-value returned from this test will be conservative because you are testing on the boundary of the parameter space."

7.22 Time Series: basics (31/12/2001)

You have to load the ts library! try

```
library(ts)
example(acf)
```

There are other packages with time series components. Try

```
help.search("time")
```

to see whatever you have installed. Under my packages section of this document, I keep a running tally of packages that have time series stuff in them

Many time series procedures want to operate on a "time series" object, not raw numbers or a data frame. You can create such an object for "somedataframe" by:

```
myTS <- ts(as.matrix(somedataframe), start=c(1992,1), frequency=12)
```

Note start and frequence designate time attributes.

7.23 Time Series: misc examples (10/04/2001)

This uses the ts package's filter(): A simple exponential weighted moving average procedure would be

```
ewma <- function (x, lambda = 1, init = 0){
  y <- filter(lambda*x, filter=1-lambda, method="recursive", init=init)
  return (y)
}
```

Using ewma as a forecast function is known as exponential smoothing. Try, e.g.:

```
x <- ts(diffinv(rnorm(100)))
y <- ewma(x,lambda=0.2,init=x[1])
plot(x)
lines(y,col="red")
```

(from Adrian Tripletti)

7.24 Categorical Data and Multivariate Models (04/25/2004)

For a dichotomous dependent variable, use the glm function in R:

```
aLogitModel <- glm(yx, family=binomial(link=logit))
```

or

```
aProbitModel <- glm(yx, family=binomial(link=probit))
```

See Jim Lindsey's package "ordinal".

And also Proportional Odds Logistic Regression in the MASS package

The VGAM package has a variety of these models: http://www.stat.auckland.ac.nz/~yee/VGAM/.

7.25 Lowess. Plot a smooth curve (04/25/2004)

library(modreg)

Users report confusion about the difference between lowess() in S and loess in R. Martin Maechler advised us, "loess() by default is not resistant/robust where as lowess() is. ... I would recommend using loess(....., family ="sym")"

R-base has "lowess"

The package "modreg" has "loess". The latter is a linear modeling-alike tool,and it can be used with other R-base tools for dealing with linear models, like the residuals function:

```
> l <- loess(y1~x1)
> plot(l)
> r <- residuals(l)
```

r is now a variable holding residuals.

7.26 Hierarchical/Mixed linear models. (06/03/2003)

User asked to estimate what we call a "contextual regression," a model of individuals, many of whom are in the same subsets (organizations, groups, etc). This is a natural case for a mixed regression model

Jose C. Pinheiro and Douglas M. Bates, Mixed Effect Models in S and S-Plus (Springer,2000).

The nlme package for R has a procedure lme that estimates these models.

In MASS, one can find the glmmPQL procedure, which can fit generalized linear models with random parameters. It repeatedly calls the lme function from the nlme package to estimate the models.

See also Jim Lindsey's repeated measurements package (available at `http://alpha.luc.ac.be/~lucp0753/rcode.html`).

7.27 Robust Regression tools (07/12/2000)

Robust Regression is available in S, its name is rreg.

There's better robust regression in the MASS package.

```
>library(MASS)
>help(rlm)
>library(lqs)
>help(lqs)
(from Thomas Lumley)
```

You could try function rlm in package MASS (in the VR bundle). That has an MM option, and was originally written to work around problems with the (then) version of lmRobMM in S-PLUS. It uses lqs in package lqs, and that is substantailly faster than the ROBETH-based code (as I started with that route). (from Brian D. Ripley)

7.28 Durbin-Watson test (10/04/2001)

The car and lmtest packages both have functions for Durbin-Watson tests. (John Fox)

7.29 Censored regression (04/25/2004)

You can do a censored regression with the survival5 package

```
library(survival5)
survreg(Surv(y,c), dist="normal")
```

where y is the censored dep-var and c is 1 if observed and 0 otherwise. (from Dan Powers)

8 Packages

8.1 What packages are installed on Paul's computer?

In the old days, when there were 200 packages, I could know what they were for and say "get this" and "get that". Now there are 3,600 and obviously that's crazy. I just have to lead by example. In addition to the recommended packages that are delivered with R, I have, as of 2012-02-13.

```
> .packages(all=TRUE)
  [1] "AER"           "animation"     "ape"
  [4] "apsrtable"     "bayesm"        "bdsmatrix"
  [7] "betareg"       "Biobase"       "BiocInstaller"
 [10] "brew"          "BRugs"         "cacher"
 [13] "car"           "CodeDepends"   "combinat"
 [16] "corpcor"       "cubature"      "debug"
 [19] "descr"         "Devore5"       "dichromat"
 [22] "digest"        "drm"           "dse"
 [25] "dse2"          "Ecdat"         "eha"
 [28] "emplik"        "ENmisc"        "EvalEst"
 [31] "evaluate"      "faraway"       "feature"
```

```
 [34] "fields"          "FlexParamCurve"  "flowCore"
 [37] "flowViz"         "fmsb"            "foreach"
 [40] "fortunes"        "frailtypack"     "gdata"
 [43] "gee"             "ggm"             "glmc"
 [46] "glmmBUGS"        "glmmLasso"       "gnm"
 [49] "gRbase"          "gvlma"           "gWidgetstcltk"
 [52] "hexbin"          "igraph"          "ISwR"
 [55] "itertools"       "ks"              "laeken"
 [58] "languageR"       "latticeExtra"    "lawstat"
 [61] "lmSupport"       "lmtest"          "lpSolve"
 [64] "ltm"             "MatrixModels"    "maxLik"
 [67] "MBESS"           "MCMCglmm"        "memoise"
 [70] "MEMSS"           "minqa"           "miscTools"
 [73] "mlmRev"          "mlogit"          "MNP"
 [76] "mosaic"          "mprobit"         "msm"
 [79] "munsell"         "mutatr"          "mvbutils"
 [82] "mvProbit"        "nls2"            "nnls"
 [85] "np"              "numDeriv"        "OrdFacReg"
 [88] "ordinal"         "orthopolynom"    "PASWR"
 [91] "pequod"          "plm"             "plyr"
 [94] "polynom"         "ppcor"           "pracma"
 [97] "pROC"            "proftools"       "pspline"
[100] "psych"           "qvcalc"          "R2OpenBUGS"
[103] "R2WinBUGS"       "RANN"            "RBGL"
[106] "rbugs"           "Rcpp"            "Rd2roxygen"
[109] "relaimpo"        "reshape2"        "RGtk2Extras"
[112] "richards"        "rlecuyer"        "robCompositions"
[115] "rockchalk"       "roxygen2"        "rsprng"
[118] "RSVGTipsDevice"  "sampling"        "scales"
[121] "SciViews"        "sem"             "SemiParBIVProbit"
[124] "setRNG"          "shape"           "simFrame"
[127] "SoDA"            "spam"            "spdep"
[130] "statmod"         "stringr"         "survey"
[133] "svDialogs"       "svGUI"           "svIDE"
[136] "svMisc"          "svSocket"        "svSweave"
[139] "svTools"         "svUnit"          "svWidgets"
[142] "tcltk2"          "tensorA"         "testthat"
[145] "tframe"          "tonymisc"        "tweedie"
[148] "ucminf"          "Zelig"           "acepack"
[151] "actuar"          "akima"           "anchors"
[154] "aod"             "arm"             "cacheSweave"
[157] "Cairo"           "CarbonEL"        "coin"
[160] "colorspace"      "CompQuadForm"    "DBI"
[163] "Deducer"         "degreenet"       "deldir"
[166] "devtools"        "diptest"         "distr"
[169] "distrEx"         "e1071"           "effects"
[172] "ellipse"         "ergm"            "FAiR"
[175] "fdrtool"         "filehash"        "flexmix"
[178] "formatR"         "Formula"         "gam"
[181] "gclus"           "geoR"            "geoRglm"
[184] "getopt"          "ggplot2"         "GPArotation"
[187] "gpclib"          "graph"           "gsubfn"
[190] "gWidgets"        "gWidgetsRGtk2"   "hergm"
[193] "HH"              "highlight"       "HSAUR"
[196] "iplots"          "ipred"           "JavaGD"
[199] "JGR"             "Kendall"         "kernlab"
[202] "latentnet"       "lavaan"          "leaps"
[205] "locfit"          "logspline"       "maptools"
[208] "matrixcalc"      "mclust"          "mda"
[211] "memisc"          "mgcv"            "mi"
[214] "mice"            "mitools"         "mix"
[217] "mlbench"         "modeltools"      "mvtnorm"
[220] "network"         "networksis"      "nor1mix"
```

```
[223] "nws"                "optparse"           "parser"
[226] "party"              "PBSmapping"         "pcaPP"
[229] "pgfSweave"          "polycor"            "proto"
[232] "qgraph"             "quantreg"           "R2STATS"
[235] "R2SWF"              "RandomFields"       "randomForest"
[238] "RArcInfo"           "Rcsdp"              "RCurl"
[241] "relevent"           "reshape"            "rgenoud"
[244] "Rglpk"              "Rgraphviz"          "RGtk2"
[247] "rJava"              "rmeta"              "rms"
[250] "roxygen"            "rpanel"             "rrcov"
[253] "RSQLite"            "rstream"            "RUnit"
[256] "scatterplot3d"      "segmented"          "sfsmisc"
[259] "shapes"             "slam"               "sna"
[262] "snow"               "snowFT"             "sp"
[265] "SparseM"            "spatstat"           "splancs"
[268] "startupmsg"         "stashR"             "statnet"
[271] "SweaveListingUtils" "systemfit"          "TeachingDemos"
[274] "tikzDevice"         "tripack"            "trust"
[277] "vcd"                "XML"                "abind"
[280] "Amelia"             "bitops"             "cairoDevice"
[283] "caTools"            "chron"              "coda"
[286] "date"               "Design"             "eco"
[289] "evd"                "fBasics"            "fGarch"
[292] "gmodels"            "gplots"             "gtools"
[295] "Hmisc"              "iterators"          "lme4"
[298] "mapproj"            "maps"               "MCMCpack"
[301] "misc3d"             "multcomp"           "multicore"
[304] "plotrix"            "polspline"          "Rcmdr"
[307] "RColorBrewer"       "relimp"             "rggobi"
[310] "rgl"                "rjags"              "rkward"
[313] "rkwardtests"        "robustbase"         "ROCR"
[316] "sandwich"           "sm"                 "strucchange"
[319] "timeDate"           "timeSeries"         "tkrplot"
[322] "urca"               "VGAM"               "xtable"
[325] "zoo"                "base"               "boot"
[328] "class"              "cluster"            "codetools"
[331] "compiler"           "datasets"           "expm"
[334] "foreign"            "graphics"           "grDevices"
[337] "grid"               "KernSmooth"         "lattice"
[340] "MASS"               "Matrix"             "methods"
[343] "nlme"               "nnet"               "parallel"
[346] "rpart"              "spatial"            "splines"
[349] "stats"              "stats4"             "survival"
[352] "tcltk"              "tools"              "utils"
```

8.2 Install and load a package

```
install.packages("lmtest", dep=c("Depends"))
```

That installs "lmtest" to the default library folder. The dep argument is for dependencies. It says I want additional packages on which lmtest depends. If you also want packages that lmtest suggests, just run it with dep=c("Depends","Suggests"). If you want everything linked to lmtest, dep=T gives the largest selection.

If you run that as an administrator, it will install packages to your main R installation's library folder. If you are not, R may check to see if it can find a place for packages in your own user folders.

8.3 List Loaded Packages

```
search()
```

displays packages that have been loaded by library() or require() as well as data frames that are attached.

8.4 Where is the default R library folder? Where does R look for packages in a computer?

Depending on your setup, R can check in many locations when you ask to load a package. Run ".libPaths()", including the period at the beginning. Here's what I see.

```
> .libPaths()
[1] "/home/pauljohn/R/x86_64-pc-linux-gnu-library/2.14"
[2] "/usr/local/lib/R/site-library"
[3] "/usr/lib/R/site-library"
[4] "/usr/lib/R/library"
```

The separate paths are useful because different install methods can be used in a system and this keeps them separate. If I'm not the administrator, they go into my home folder, as you see above in item [1]. The fourth folder in the list is the one that R provided when it was installed. On Debian Linux, the path [2] is used for packages that are delivered as Debian "deb" packages.

If you only see one path in the output from your computer, it probably means you need to do some system administration work to add more paths. Read ?.libPaths. On my system, those paths are set in an environment file, /etc/R/Renviron.

8.5 Detach libraries when no longer needed (10/04/2001)

Use detach(package:...), for instance:

```
> search()
[1] ".GlobalEnv"    "package:ctest" "Autoloads"     "package:base"
> detach(package:ctest)
> search()
[1] ".GlobalEnv"    "Autoloads"     "package:base"
(from Emmanuel Paradis)
```

9 Misc. web resources

9.1 Navigating R Documentation (12/02/2012)

R is delivered with a set of manuals

```
help.start()
```

They are also available online. Go to any CRAN mirror site (follow CRAN link on `http://www.r-project.org`) . Choose a mirror, look for "Documentation" on the left side of the page. The "Manuals" and "FAQs" links point to the same materials that are delivered with R itself.

There are also contributed docs, which are donated by the community: http://cran.r-project.org/other-docs.html. Where once we (R users) had no introductory material at all, there is now a profusion, as it appears at least 50 different people have written a "Beginners guide to R".

9.2 R Task View Pages (12/02/2012)

For me, navigating the R web material is frustrating because I never find things in the same way twice. You are unlikely to find these, unless you know they are there.

Task Views are written by the experts, who survey what packages are available. On any CRAN mirror, they are in a folder web/views, as in: `http://cran.r-project.org/web/views`. Many

of these Task View essays are absolutely fabulous! I can't say strongly enough my admiration to the authors who have obviously poured a massive effort into preparing concise and comprehensive reviews.

9.3 Using help inside R(13/08/2001)

help(function) same as ?function

help.search("something") searches in all available functions from all loaded packages for the word "something".

Don't forget that help.search takes regular expressions (as its help describes)

```
R> help.search(".*trans.*")
Help files with name or title matching `.*trans.*':

boxcox(MASS)              Box-Cox Transformations for Linear Models
gilgais(MASS)             Line Transect of Soil in Gilgai Territory
heart(MASS)               The Stanford heart transplant data
(Thomas Lumley)
```

There is also an "apropos" function, similar in nature to emacs. If you have a vague idea of what you want, try it. Want to find how to list files? Do:

```
apropos("files")
```

Often you don't know the keyword to look for.

In my opinion, by far the best feature introduced in the last few years is RSiteSearch. A command like

```
RSiteSearch("logistic regression")
```

Will search in an online system. It brings up a web page, showing that you have searched in package function listings. Adjust the radio dial to add the search of the r-help email list, and you will really make some progress.

9.4 Run examples in R (10/04/2001)

See "example" at end of output in help?

Try

```
> ?ls
> example(ls)
> example(lm)
> example(mean)
```

Graphs go by too fast? The default in R is now set like this:

par(ask=TRUE)

so that graphs will not be drawn without your permission.

10 R workspace

10.1 Writing, saving, running R code (31/12/2001)

The easiest thing is to use some editor, Emacs or Winedit, write your R code, and then run it into R, and if it fails, change it in the editor, run it it again. Emacs with ESS installed will make this work fine. I've heard of plugins for other editors, did not try them, though.

If you don't like that, you can type commands directly into the R window and they have some handy things.

Up and down arrows cycle through the history of commands.

If you type in a function and want to revise it:

fix(f) or g <- edit(f) should bring up an editor.

I personally don't see any point in this, though, since you should just use an editor.

10.2 .RData, .RHistory. Help or hassle? (31/12/2001)

When you quit R, since version 1 or so, it asks if you want to save the workspace. If you say yes, it creates a file .RData. The next time you start R in this working directory, it opens all those objects in .RData again. Type "objects()" to see.

If you want that, OK. If you don't, say NO at quit time, or erase the .RData file before starting R.

10.3 Save & Load R objects (31/12/2001)

If you try

```
> a <- c(1,2,3)
> save(a, file="test.RData")
> rm(a)
> load("test.RData")
> a

[1] 1 2 3
```

(from Peter Dalgaard, 12/29/2001)

10.4 Reminders for object analysis/usage (11/08/2000)

```
objects() # lists all objects
attributes(anObject)
str(anObject) # diagnostic info on anObject
page(anObject) # pages through anObject
rm(anObject) # removes an object
rm(list=ls()) # removes all objects found by ls()
```

10.5 Remove objects by pattern in name (31/12/2001)

To better understand this, do a Web search for "regular expression matching." Suppose you want to remove objects beginning with lm (lm1, lm2, lm3 etc).

```
rm(list=objects(pattern="^lm.*"))
(from Kjetil Kjernsmo)
```

Don't do this to remove all objects with names like g01, g02, etc:

```
rm(list=ls(pat="g*"))
```

That removes everything! To remove objects whose names have "g" anywhere in the string, use

```
rm(list=ls(pat="g"))
```

, or names starting with "g" one can use

```
rm(list=ls(pat="^g"))
```

The caret "^" and dollar sign "$" are regex symbols tha represent the beginning and ending of a string.

10.6 Save work/create a Diary of activity (31/12/2001)

consider sink("filename")

The file .Rhistory contains commands typed on Unix systems.

Emacs users can run R in a shell and save everything in that shell to a file.

Ripley added: But if you are on Windows you can save the whole console record to a file, or cut-and-paste sections. (And that's what I do on Unix, too.)

10.7 Customized Rprofile (31/12/2001)

In your HOME directory, the file ~/.Rprofile. For example, if you think the stars on significance tests are stupid, follow Brian Ripley's example:

```
options(show.signif.stars=FALSE)
ps.options(horizontal=FALSE)
```

There is a system-wide Rprofile you could edit as well in the distribution

11 Interface with the operating system

11.1 Commands to system like "change working directory" (22/11/2000)

Commands to the operating system: system("insert command here between quotes").

For the particular case of changing directories use getwd/setwd [I think system() spawns its own shell, so changing directories with system() isn't persistent] . For example:

```
> getwd()    ## start in my home directory
[1] "/home/ben"
> system("cd ..")  ## this executes OK but ...
> getwd()
[1] "/home/ben"     ## I'm still in the same working directory
> setwd("..")       ## this is the way to change working directory
NULL
> getwd()           ## now I've moved up one
[1] "/home"
```

(from Ben Boker)

11.2 Get system time. (30/01/2001)

```
> date()
[1] "Tue Jan 30 10:22:39 2001"
> Sys.time()
[1] "2001-01-30 10:21:29"
> unclass(Sys.time())
[1] 980871696
```

This is the number of seconds since the epoch (Jan 1, 1970). I use it as a "time stamp" on files sometimes.

11.3 Check if a file exists (11/08/2000)

The function file.exists() will do this

11.4 Find files by name or part of a name (regular expression matching) (14/08/2001)

Do

```
list.files()
```

list.files() gives back a vector of character strings, one for each file matching the pattern. Want file names ending in .dat? Try:

```
myDat <- list.files(pattern="\\.dat$")
```

Then iterate over this myDat object, using the elements myDat[[i]] if you do something to each file.

Please note patterns in R are supposed to be regular expressions, not ordinary shell wildcards. Regular expressions are fairly standard in unix, but each version has its wrinkles. To see about the grep command, or about apropos, where regexp is used, do:

```
?grep
or
?apropos
```

Martin Maechler spelled out a few of the basics for me:

```
"."  means  ``any arbitrary character''
"*"  means  ``the previous [thing] arbitrary many (0,1,...) times''
             where [thing] is a single character (or something more general
                                                  for advanced usage)
```

12 Stupid R tricks: basics you can't live without

12.1 If you are asking for help (12/02/2012)

1. With your question, include the code you run that produces the problem you are asking about. Clean up your code, make a "minimial working example" that displays the problem. You are much more likely to get a reasonable answer if you ask the question completely.
2. With your question, include the output from sessionInfo(). ALWAYS include that in any message you send to ask for help from anyone, including me, r-help email list, stack overflow, or whatever.

```
> sessionInfo()
R version 2.14.1 (2011-12-22)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
[1] tools_2.14.1
```

12.2 Commenting out things in R files (15/08/2000)

Put ## at beginning of line, or

"With a good code editor none of these are any problem. There is no reason to write many lines of comments in R code: that's what the .Rd file is for. And if you want to temporarily delete a block, if(0) { ... } will work, as indeed will temporarily deleting in an editor." (from Brian D. Ripley)

Don't forget Emacs with ESS has the ability to comment-out a highlighted region:

M-x comment-region

or, more simply, highlight a section and type M-; (that's alt and semicolon on most systems).

13 Misc R usages I find interesting

13.1 Character encoding (01/27/2009)

With increasing frequency, I have noticed the problem that when I copy text from R, it shows up in the output with noise characters, like a quotation mark that shows up as *&\ or whatever. It turns out this is a matter of "character encoding", and in the r-help list on 2009–01–27, Duncan Murdoch made an absolutely beautiful post about it.

```
This looks like an encoding problem:  there are several different standards for
    encoding non-ASCII characters.  All of your tools have to agree on the
    encoding.
To my eye it looks as though in the first case R is writing out UTF-8, and
    whatever you are using to look at your .tex file is assuming latin1 (some
    Windows programs say "ANSI", but I think that doesn't fully specify the
    encoding:  you also need a code page, which is set somewhere in Windows
    control panel.)
The functions related to encodings in R are:
 options(encoding="latin1")  - set the default encoding
 iconv(x, from="latin1", to="UTF-8")  - re-encode entries, mapping each character
      from one encoding to the other
 Encoding(x) - display the encoding of each entry (unknown means ascii or the
     native encoding for your platform)
 Encoding(x) <- "latin1" - change the declared encoding, without changing the
     bytes.
```

13.2 list names of variables used inside an expression (10/04/2001)

```
> all.names(expression(sin(x+y)))
[1] "sin" "+"   "x"   "y"
> all.names(functions=FALSE,expression(sin(x+y)))
[1] "x" "y"
> all.vars(functions=FALSE,expression(sin(x+y)))
[1] "x" "y"
> all.vars(expression(sin(x+y)))
[1] "x" "y"
```

all.vars() works on a formula, terms, or expression object.

```
> all.vars((~j(pmin(3,g(h(x^3*y))))))
[1] "x" "y"
```

13.3 R environment in side scripts (10/04/2001)

The command

```
rm(list=ls())
```

removes all "visible" variables and functions from the workspace, but it leaves behind variables that start with period ("dot" files) and variables in environment that begin with dot.

This function does not remove objects correctly if they are hidden in .GlobalEnv

```
> testLoadSeveralHDF <- function(numFiles) {
>   for (i in 0:(numFiles-1)) {
>     filename <- paste("trial",i,".hdf",sep="")
>     print(filename)
>     hdf5load(filename)
>     rm(list = ls(pat="^g"))
>   }
> }
```

Doug Bates said:

Add envir = .GlobalEnv to both the ls and the rm calls. That is

```
rm(list = ls(pat = ^g, envir = .GlobalEnv), envir = .GlobalEnv)
```

13.4 Derivatives (10/04/2001)

first and second derivative of the expression wrt x

```
>D(expression(x^2),"x")
>D(D(expression(x^2),"x"),"x")
```

The real R FAQ 7.6 explains "How can I get eval() and D() to work?"